AMEIR: Automatic Behavior Modeling, Interaction Exploration and MLP Investigation in the Recommender System

Pengyu Zhao[*1,2], **Kecheng Xiao**[*1], **Yuanxing Zhang**[*3], **Kaigui Bian**[1] and **Wei Yan**[1]
[1]School of EECS, Peking University, Beijing, China
[2]HULU LLC, Beijing, China
[3]Alibaba Group, Beijing, China
{pengyuzhao, kecheng, bkg, w}@pku.edu.cn, yuanxing.zyx@alibaba-inc.com

Abstract

Recently, deep learning models have been widely explored in recommender systems. Though having achieved remarkable success, the design of task-aware recommendation models usually requires manual feature engineering and architecture engineering from domain experts. To relieve those efforts, we explore the potential of neural architecture search (NAS) and introduce AMEIR for **A**utomatic behavior **M**odeling, interaction **E**xploration and multi-layer perceptron (MLP) **I**nvestigation in the **R**ecommender system. Specifically, AMEIR divides the complete recommendation models into three stages of behavior modeling, interaction exploration, MLP aggregation, and introduces a novel search space containing three tailored subspaces that cover most of the existing methods and thus allow for searching better models. To find the ideal architecture efficiently and effectively, AMEIR realizes the one-shot random search in recommendation progressively on the three stages and assembles the search results as the final outcome. The experiment over various scenarios reveals that AMEIR outperforms competitive baselines of elaborate manual design and leading algorithmic complex NAS methods with lower model complexity and comparable time cost, indicating efficacy, efficiency, and robustness of the proposed method.

1 Introduction

Recommender system has become an essential service on online E-commerce business and content platforms to deliver items that best fit users' interests from the substantial number of candidates. Recently, advancements of deep learning (DL) have innovated the recommending strategies in real-world applications [Covington *et al.*, 2016; Zhou *et al.*, 2018] to enable accurate and personalized recommendation.

DL-based recommender systems have to process both sequential and non-sequential input features, and thereby follow a characterized architecture compared to the general DL tasks. A canonical DL-based recommendation model usually includes three parts: (1) **Behavior modeling** [Hidasi and Karatzoglou, 2018; Zhou *et al.*, 2018] probes into capturing user's diverse and dynamic interest from historical behaviors; (2) **Interaction exploration** [Cheng *et al.*, 2016; Guo *et al.*, 2017] finds useful interactions among different fields to provide memorization and intuitive conjunction evidence for the recommendation. (3) **Multi-layer perceptron (MLP) investigation** [Covington *et al.*, 2016] aggregates the inputs with the results from the previous parts, and then fuses the features with hidden layers in the MLP.

Industry demands efficiently designing the three parts in the DL-based recommendation model to improve the accuracy and personalization for given recommender tasks in business. Nevertheless, it is hard to find a unified model meeting the requirements in all the scenarios. For the models of **behavior modeling**, recurrent neural network (RNN) [Hidasi and Karatzoglou, 2018] is hard to preserve long-term behavioral dependencies even though employing gated memory cells. Convolutional neural network (CNN) [Yuan *et al.*, 2019] is capable of learning local feature combinations yet it relies on a wide kernel or deep layers to distinguish long-term interests. The attention mechanism [Zhou *et al.*, 2018] directly aggregates the entire behavior sequence, but it can not capture the evolution of the user's preference [Zhou *et al.*, 2019]. Self-attention [Vaswani *et al.*, 2017] is better for modeling long-term behaviors [Feng *et al.*, 2019; Sun *et al.*, 2019], but it is hard to be deployed in real-time applications that require fast inference with limited resources. Regarding the methods for **interaction exploration** and **MLP investigation**, vanilla MLP [Covington *et al.*, 2016] implicitly generalizes high-order information in the network. To help memorize useful features from raw inputs, the existing literature extracts implicit and explicit low-order [Cheng *et al.*, 2016; Guo *et al.*, 2017] and high-order [Lian *et al.*, 2018] interactions and combine them with a pre-defined MLP. However, these methods either require hand-crafted cross features or simply enumerate interactions of bounded degree, and thus introducing noise in the model [Liu *et al.*, 2020]. Moreover, the MLP specified manually or by grid search is usually sub-optimal. Hence, adopting and coordinating the task-aware architectures for those **three parts** are the main challenge for building **efficient** and **accurate** recommender systems.

Recently, the neural architecture search (NAS) paradigm

[*]Equal Contribution

[Zoph and Le, 2017; Real *et al.*, 2019] is proposed to automatically design deep learning models by searching task-specific and data-oriented optimal architecture from the search space, thereby mitigating a lot of human efforts. Some latest attempts [Luo *et al.*, 2019; Liu *et al.*, 2020; Song *et al.*, 2020] incorporate NAS with the recommendation. Nevertheless, they all neglect **behavior modeling** and only consider **interaction exploration** and **MLP investigation** in the **restricted** search spaces. Different from these methods, we introduce **AMEIR**, namely **A**utomatic behavior **M**odeling, interaction **E**xploration and MLP **I**nvestigation in the **R**ecommender system, to **automatically** find the **complete** recommendation models for adapting **various** recommender tasks and mitigating manual efforts. Specifically, AMEIR divides the backbone recommendation models based on the three stages of behavior modeling, interaction exploration, MLP aggregation, and introduces a novel search space containing three tailored subspaces that are designed to cover most of the representative recommender systems and thus allow for searching better architectures. Facing the industrial demands of agile development and architecture iteration, AMEIR realizes the efficient while competitive **one-shot random search** [Li and Talwalkar, 2019], and proposes a three-step searching pipeline to **progressively** find the ideal architectures for the corresponding three stages. The search results will be assembled as the final outcome. The experimental results over various recommendation scenarios show that our automatic paradigm consistently achieves state-of-the-art performance despite the influence of multiple runs, and could outperform both the strongest baselines with elaborate manual design and leading NAS methods with comparable search cost, indicating efficacy, efficiency, and robustness of the proposed method. Further analysis reveals that the overall time cost of AMEIR is similar to the baseline methods without architecture search and the searched models always possess lower model complexity, which demonstrates that AMEIR is efficient in both search and inference phases.

2 AMEIR

In Sec.2, we will introduce the three-stage search space, and the tailored three-step one-shot searching pipeline in AMEIR. Finally, we will discuss the relationship between AMEIR and representative recommender systems.

2.1 Backbone Model in AMEIR's Search Space

The recommendation models in AMEIR's search space share the same three-stage backbone, as illustrated in Fig.1.

Stage 0: Feature Representation and Embedding. In the common recommender systems, the input features are collected in multi-field categorical form, and segregated into sequential features and non-sequential features: (1) The sequential features describe the user's behaviors, which can be represented by a sequence of behavior elements. Each behavior element contains the item profile and the historical context profile. (2) The non-sequential features depict the attribute information of the current recommendation, including user profile, instant context profile, and the optional target profile for click-through rate (CTR) prediction task.

Conforming to [Covington *et al.*, 2016; Zhou *et al.*, 2018], AMEIR transforms the data instances into high-dimensional sparse vectors via one-hot or multi-hot encoding. Each data instance can be formally represented by $\mathbf{x} = [\mathbf{x}^a, \mathbf{x}^b] = [\mathbf{x}_1^a, \mathbf{x}_2^a, ..., \mathbf{x}_T^a, \mathbf{x}_1^b, \mathbf{x}_2^b, ..., \mathbf{x}_N^b]$. $\mathbf{x}^a = [\mathbf{x}_1^a, \mathbf{x}_2^a, ..., \mathbf{x}_T^a]$ denotes the encoded sequential behavior elements of length T with each behavior element $\mathbf{x}_i^a$ grouped in N_a categorical fields of the same dimension, and $\mathbf{x}^b = [\mathbf{x}_1^b, \mathbf{x}_2^b, ..., \mathbf{x}_N^b]$ represents N encoded vectors of non-sequential feature fields. AMEIR compresses the sparse features into low dimensional dense vectors via embedding technique. Suppose there are F unique features. AMEIR creates the embedding matrix $\mathbf{E} \in \mathbb{R}^{F \times K}$, where each embedding vector has dimension K. The embedded data instance can then be represented by $\mathbf{e} = [\mathbf{e}^a, \mathbf{e}^b] = [\mathbf{e}_1^a, \mathbf{e}_2^a, ..., \mathbf{e}_T^a, \mathbf{e}_1^b, \mathbf{e}_2^b, ..., \mathbf{e}_N^b]$.

Stage 1: Behavior Modeling on Sequential Features. To extract the sequential representation from user's behavior, AMEIR introduces a **block-wise behavior modeling network** to model sequential patterns based on grouped behavioral embedding $\mathbf{H}_0^a = [\mathbf{e}_1^a, \mathbf{e}_2^a, ..., \mathbf{e}_T^a]$. The behavior modeling network is constructed by stacking a fixed number of L_a residual **building blocks**, where the l-th block receives the hidden state $\mathbf{H}_{l-1}^a$ from the previous block as input and generates a new hidden state $\mathbf{H}_l^a$ of the same dimension. Each block consists of three consecutive operations of normalization, layer, and activation, which are selected from the corresponding **operation sets** in the **search subspace of building blocks**. The hidden state of layer l is formulated as:

$$\mathbf{H}_l^a = \text{Act}_l^a(\text{Layer}_l^a(\text{Norm}_l^a(\mathbf{H}_{l-1}^a))) + \mathbf{H}_{l-1}^a, \qquad (1)$$

where Act_l^a, Layer_l^a and Norm_l^a are the selected activation, layer and normalization operations for layer l. As the fully connected networks can only handle fixed-length inputs, AMEIR compresses the final hidden states into fixed dimensions via sequence-level sum pooling: $\mathbf{h}_{pool}^a = \sum_{i=1}^{T} \mathbf{h}_{L_a,i}^a$, where $\mathbf{h}_{L_a,i}^a$ denotes the i-th element in $\mathbf{H}_{L_a}^a$. Moreover, to extract the latest interest of the user, the last hidden state $\mathbf{h}_{L_a,T}^a$ is also included in the result. Besides, when the target item is involved, AMEIR adopts an attention layer to extract a target-aware representation: $\mathbf{h}_{att}^a = \sum_{i=1}^{T} a_i \mathbf{h}_{L_a,i}^a$, with a_i computed by the local activation unit [Zhou *et al.*, 2018]. The concatenation $\mathbf{h}^a = [\mathbf{h}_{pool}^a, \mathbf{h}_{L_a,T}^a, \mathbf{h}_{att}^a]$ is served as the sequential feature extracted from Stage 1. In practice, this stage remains the procedure to execute looking up embedding only once, which is orthogonal to the industrial large-scale sparse-features lookup optimizations.

Stage 2: Interaction Exploration on Non-sequential Features. Learning effective feature interactions is crucial for recommender systems. Different from the widely-used methods that manually extract useful interactions or exhaustively enumerate interactions of bounded degrees [Cheng *et al.*, 2016; Guo *et al.*, 2017; Lian *et al.*, 2018], AMEIR explores a small number of M beneficial interactions $\mathbf{p}^b = [\mathbf{p}_1^b, \mathbf{p}_2^b, ..., \mathbf{p}_M^b]$ from the **search subspace of interactions**. The search subspace contains all low-order and high-order interactions among the non-sequential features, where each interaction has the same dimension as the input embeddings.

Stage 3: Aggregation MLP Investigation with Shortcut Connection. Given the inputs of sequential feature,

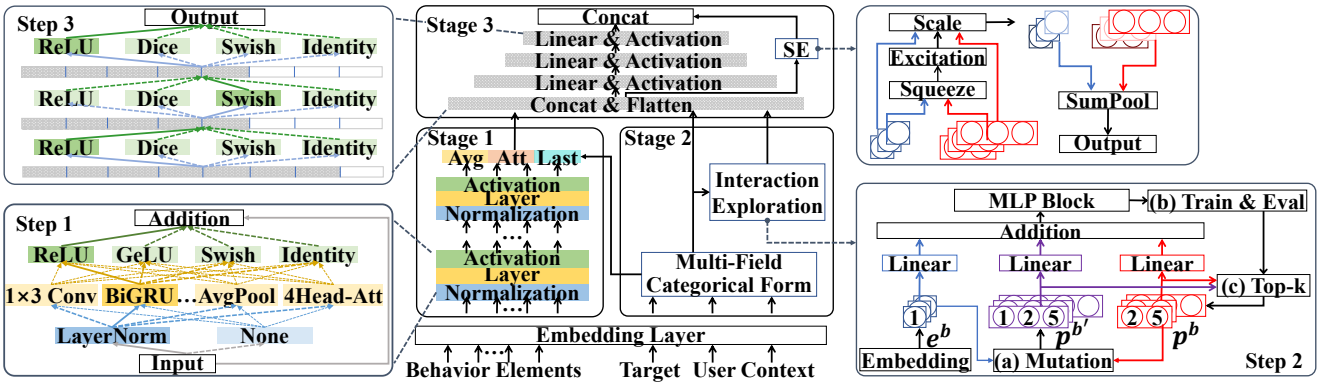

Figure 1: The three-stage search space and the three-step one-shot searching pipeline of AMEIR. AMEIR searches for building blocks, feature interactions, and aggregation MLP (three layers as an example) from corresponding search subspaces in Steps 1-3.

non-sequential features, and explored interactions $\mathbf{h}_0^c = [\mathbf{h}^a, \mathbf{e}^b, \mathbf{p}^b]$, AMEIR aggregates these features by taking advantage of both generalization from MLP and memorization from a shortcut connection. As MLP affects both recommendation quality and efficiency, AMEIR searches for the MLP settings in Stage 3. Assume the aggregation MLP has L_c layers. AMEIR chooses the **dimension** and **activation function** for each layer from the **search subspace of MLP**. The forward pass of each hidden layer l can be depicted as:

$$\mathbf{h}_l^c = \text{Act}_l^c(\mathbf{W}_l^c \mathbf{h}_{l-1}^c + \mathbf{b}_l^c), \tag{2}$$

where $\mathbf{h}_l^c$, $\mathbf{W}_l^c$, $\mathbf{b}_l^c$ and Act_l^c are the output, weight, bias and activation function of the l-th layer. Meanwhile, inspired by the bypaths in [Cheng *et al.*, 2016; Guo *et al.*, 2017; Lian *et al.*, 2018], AMEIR introduces a **Squeeze-and-Excitation** (SE) [Hu *et al.*, 2018] shortcut connection to introduce attentive recalibration on the feature embeddings and explored interactions $\mathbf{Q}^{se} = [\mathbf{e}^b, \mathbf{p}^b] = [\mathbf{e}_1^b, ..., \mathbf{e}_N^b, \mathbf{p}_1^b, ..., \mathbf{p}_M^b]$. The attention units $\mathbf{a}^{se} = [a_1^{se}, a_2^{se}, ..., a_{M+N}^{se}]$ are computed by $\mathbf{a}^{se} = \sigma(\mathbf{W}_2^{se}\text{ReLU}(\mathbf{W}_1^{se}\mathbf{Q}^{se}\mathbf{v}^{se}))$, where $\mathbf{v}^{se}$ is applied for linear projection, and $\mathbf{W}_1^{se}, \mathbf{W}_2^{se}$ are weight matrices in the fully-connected layers. The output of SE connection is given by the weighted average $\mathbf{h}^{se} = \sum_{i=1}^{M+N} a_i^{se}\mathbf{q}_i^{se}$, where $\mathbf{q}_i^{se}$ is the i-th element in $\mathbf{Q}^{se}$. AMEIR concatenates the results from aggregation MLP and SE connection as the output representation $\mathbf{h}^c = [\mathbf{h}_{L_c}^c, \mathbf{h}^{se}]$.

Loss Function. For the CTR prediction task, AMEIR minimizes the binary log loss for optimization:

$$L = -\frac{1}{|\mathcal{S}|}\sum_{(\mathbf{x},y)\in\mathcal{S}} [y\log\sigma(\mathbf{w}^\top\mathbf{h}^c) + (1-y)\log(1-\sigma(\mathbf{w}^\top\mathbf{h}^c))],$$

where $\sigma(\mathbf{w}^\top\mathbf{h}^c)$ is the predicted CTR score, and $\mathcal{S}$ is the training set with $\mathbf{x}$ as input and $y \in \{0, 1\}$ as label. For the item retrieval, we adopt the sampled cross entropy loss:

$$L = -\frac{1}{|\mathcal{S}|}\sum_{(\mathbf{x},i)\in\mathcal{S}} [\log\sigma(\mathbf{e}_i^\top\mathbf{W}\mathbf{h}^c) + \sum_{j\in\mathcal{I}_i^-}\log(1-\sigma(\mathbf{e}_j^\top\mathbf{W}\mathbf{h}^c))],$$

where $\sigma(\mathbf{e}_i^\top\mathbf{W}\mathbf{h}^c)$ is the relevance score between target item i and final representation; $\mathcal{S}$ is the training set of input feature $\mathbf{x}$ and target item i; $\mathcal{I}_i^-$ is the item set of negative samples.

2.2 Search Subspaces of Three Stages

For each stage in the backbone, AMEIR introduces a corresponding search subspace to cover the representative recommendation models. We will describe each of the three subspaces below and show a detailed comparison in Sec.2.4.

Search Subspace of Building Blocks. In order to cover the existing behavior modeling methods, AMEIR proposes a **search subspace of building blocks** in the behavior modeling network, including three operation sets for normalization, layer, and activation. Specifically, AMEIR collects the **normalization set** of {Layer normalization [Ba *et al.*, 2016], None} and **activation set** of {ReLU, GeLU [Vaswani *et al.*, 2017], Swish [Ramachandran *et al.*, 2017], Identity} that are commonly used in the previous methods. Regarding the layer operations, AMEIR introduces four categories of candidate layers, namely **convolutional layers**, **recurrent layers**, **pooling layers**, and **attention layers** to identify sequential patterns in the user history. The convolutional layers are all one-dimension, including standard convolution with kernel size {1, 3} and dilated convolution with kernel size {3, 5, 7}. The pooling layers consist of average and max poolings with kernel size 3. Both convolutional and pooling layers are of stride 1 and SAME padding. Bi-directional GRU (Bi-GRU) is employed as the recurrent layer as it is faster than Bi-LSTM without loss of precision. Two-head and four-head bi-directional self attentions [Vaswani *et al.*, 2017] are also introduced for better behavior modeling. Additionally, when the target item appears in the input, the layer operation set will associate an attention layer that attends to each sequence position from target [Zhou *et al.*, 2018]. Besides, **zero operation** is also included to implicitly allow a dynamic depth of behavior modeling network. It is worth emphasizing that the operation sets are proposed to cover the **existing** methods. When a new operation is developed, it can be easily added to the operation set to find better recommendation models.

Search Subspace of Interactions. The existing literature adopts **Hadamard product** [Lian *et al.*, 2018], **inner product** [Guo *et al.*, 2017; Qu *et al.*, 2018], **bilinear function** [Huang *et al.*, 2019] and **cross product** [Cheng *et al.*, 2016] in the feature interactions. Hadamard product calculates the element-wise product among the feature embed-

dings, which can be exploited by both MLP and shortcut connection. The inner product can be seen as a simple form of Hadamard product compressed by sum pooling. Bilinear function introduces extra parameters on Hadamard product to learn fine-grained representations, but it can not handle high-order interactions and does not exhibit superiority over Hadamard product when combining with MLP [Huang *et al.*, 2019]. Although the cross product can yield a more flexible form of interactions, it can only be used in the shortcuts [Luo *et al.*, 2019] because the parameters will exponentially explode in the order of interactions and can not be well trained due to the low frequency of occurrence. Therefore, AMEIR chooses **Hadamard product** as the interaction function due to its expressive power and flexibility. The **search subspace of interactions** includes all low-order and high-order Hadamard interactions. An interaction of order r can be expressed as $\mathbf{e}^b_{i_1} \odot \mathbf{e}^b_{i_2} \odot ... \odot \mathbf{e}^b_{i_r}$. where $\odot$ denotes Hadamard product and $\mathbf{e}^b_{i_1}, \mathbf{e}^b_{i_2}, ..., \mathbf{e}^b_{i_r}$ are selected from $\mathbf{e}^b$.

Search Subspace of MLP. The **search subspace of MLP** includes **dimensions** and **activation functions** of hidden layers. Assume K^c_0 is the dimension of $\mathbf{h}^c_0$. The dimensions of hidden layers are chosen from $\{0.1, 0.2, ..., 1.0\}$ of K^c_0 while ensuring monotonically non-increasing following the practice. The activation functions are selected from {ReLU, Swish, Identity, Dice [Zhou *et al.*, 2018]}.

2.3 Three-step One-shot Searching Pipeline

One-shot NAS in AMEIR. Recent approaches [Pham *et al.*, 2018] introduce efficient **one-shot weight-sharing** paradigm in NAS to boost search efficiency, where all **child models** share the weights of common operations in a large **one-shot model** that subsumes every possible architecture in the search space. Though diverse complex search algorithms have been coupled with one-shot NAS, the latest study [Li and Talwalkar, 2019] reveals that the efficient **one-shot random search** is surprisingly better than the complex gradient-based methods of DARTS [Liu *et al.*, 2019], SNAS [Xie *et al.*, 2019] and RL-based methods of ENAS [Pham *et al.*, 2018]. To facilitate the industrial demands of agile development and architecture iteration, AMEIR realizes the one-shot random search in the recommendation model: (1) The child models are **randomly** sampled from the search spaces (actually subspaces) to train the shared weights of the one-shot model and validate its performance on a small subset of validation instances to ensure adequate training while avoiding overfitting. (2) A subset of child models is **randomly** selected and evaluated on the same validation subset using the inherited weights from the one-shot model. (3) The child models with the best performance will be derived and retrained for the final recommendation outcome. Moreover, to reduce the magnitude of the search space and find the better recommendation models, AMEIR modularizes the search process into **three** steps matching the three stages in the backbone model by progressively searching for the architectures in the three search subspaces. Meanwhile, AMEIR ensures the fairness of one-shot model training to alleviate the representation shift caused by weight sharing. Details of the three steps are shown below.

Step 1: Behavior Modeling Search. In Step 1, AMEIR searches for the behavior modeling networks from the search subspace of building blocks, as shown in Fig.1. The normalization, layer, and activation operations are randomly selected from the operation sets to build the recommendation model. To decouple behavior modeling from other stages, a pre-defined MLP is exploited to combine $\mathbf{h}^a$ and $\mathbf{e}^b$. The training and evaluation are the same as previously described.

Step 2: Interaction Exploration. In Step 2, AMEIR combines one-shot NAS and sequential model-based optimization (SMBO) algorithm [Liu *et al.*, 2018] to explore interactions on non-sequential features. The interaction set $\mathbf{p}^b$ is initialized with non-sequential features $\mathbf{e}^b$, and then progressively updated by several cycles of evolution to increase the interaction order, as illustrated in Fig.1: (a) The candidate interactions in $\mathbf{p}^b$ intersect with all non-sequential fields from $\mathbf{e}^b$ via Hadamard product to mutate for higher-order interactions. The expanded interaction set is denoted by $\mathbf{p}^{b'}$. (b) The intersections in $\mathbf{p}^{b'}$ are trained by one-shot NAS and evaluated on the validation set. (c) Top-k interactions with the highest validation fitness are retained as the new $\mathbf{p}^b$, while other interactions are filtered out to avoid exhaustively visiting all possible solutions. After the evolution, AMEIR selects top-M features in $\mathbf{p}^b$ as the resulting interaction set. In order to train and evaluate interactions in the one-shot model, AMEIR utilizes another pre-defined MLP to aggregate interactions in $\mathbf{p}^{b'}$ with non-sequential embeddings in $\mathbf{e}^b$, while the behavior modeling here is left out for efficiency. As the linear projection of feature concatenation is equivalent to feature-wise addition of projected results, i.e., $\mathbf{W}\mathbf{h}^\mathsf{T} = [\mathbf{W}_1, \mathbf{W}_2, ..., \mathbf{W}_N][\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_N]^\mathsf{T} = \sum_{i=1}^{N} \mathbf{W}_i \mathbf{h}_i^\mathsf{T}$, the one-shot NAS in Step 2 is efficiently realized by adding the projections of one uniformly-sampled interaction and all embeddings in $\mathbf{e}^b$ as the output of the first hidden layer, without accessing to all interactions in $\mathbf{p}^{b'}$. To further speed up the search process, the weights of the one-shot model are inherited in consecutive evolution cycles.

Step 3: MLP Investigation. In Step 3, AMEIR investigates the dimensions and activations in the aggregation MLP based on the found behavior modeling network and interactions. To find the hidden sizes of MLP, AMEIR pre-allocates a weight matrix with the maximum dimension $\mathbf{W}^c \in \mathbb{R}^{K^c_0 \times K^c_0}$ for each layer. Suppose h_{in} and h_{out} are chosen to be the hidden sizes of previous and current layers. AMEIR slices out sub-matrix $\mathbf{W}^c[: h_{\text{in}}, : h_{\text{out}}]$ to assemble the MLP in Eqn. (2). Other details are similar to Steps 1 and 2.

Derivation. After the three-step random search, AMEIR assembles the search results as the final outcome and retrains the derived model for evaluation. It is worth emphasizing that the derived model is initialized with the one-shot model weights because the training protocols of model architectures and datasets are the same for both one-shot model and child models, which is in contrast to the most existing methods that adopt proxy task for one-shot model training [Liu *et al.*, 2019; Xie *et al.*, 2019]. The inherited weights accelerate the retraining process and compensate for the search cost.

2.4 Related Work to AMEIR

Behavior Modeling. The existing methods introduce RNN [Hidasi and Karatzoglou, 2018], CNN [Yuan *et al.*,

Category	Model	AUC	QPS
DNN-based	DNN [Covington *et al.*, 2016]	0.6304	64K
	Wide&Deep [Cheng *et al.*, 2016]	0.6313	55K
	DeepFM [Guo *et al.*, 2017]	0.6320	58K
Attention-based	DIN [Zhou *et al.*, 2018]	0.6341	53K
	DIEN [Zhou *et al.*, 2019]	0.6344	44K
	DSIN [Feng *et al.*, 2019]	0.6352	43K
AMEIR	AMEIR-A	0.6357 ± 0.0001	47K
	AMEIR-B	0.6359 ± 0.0003	43K
	AMEIR-C	**0.6383** ± 0.0004	46K

Table 1: Comparison with representative methods on Alimama.

Model	Beauty		Steam	
	HR@5	NDCG@5	HR@5	NDCG@5
BPR [Rendle *et al.*, 2009]	0.301	0.227	0.527	0.369
GRU4REC$^+$ [Hidasi and Karatzoglou, 2018]	0.300	0.215	0.626	0.468
NARM [Li *et al.*, 2017]	0.243	0.166	0.658	0.501
NextItNet [Yuan *et al.*, 2019]	0.334	0.243	0.679	**0.517**
BERT4REC [Sun *et al.*, 2019]	0.337	0.252	0.676	0.513
AMEIR-A	**0.341** ± 0.004	**0.257** ± 0.003	**0.681** ± 0.003	**0.517** ± 0.002

Table 2: Comparison with representative methods on Beauty/Steam.

2019], or Transformer-based [Sun *et al.*, 2019] behavior modeling networks to capture sequential relations and evolved interests from user behaviors. Then, the sequential features are extracted from the sequence-level hidden states by either taking the last hidden state [Kang and McAuley, 2018] or aggregating the sequence by a pooling layer [Cheng *et al.*, 2016; Covington *et al.*, 2016]. Some recent works [Zhou *et al.*, 2018] further employ an attention mechanism between the sequential representations and target item to retrieve the user's main proposal. These methods can all be found in AMEIR's search space. Specifically, the search subspace of building blocks contains all layer, activation, and normalization operations in those behavior modeling networks. Moreover, the last hidden state, pooling-based aggregation, and attention-based aggregation are all used for sequence compression.

Interaction Exploration. In parallel with behavior modeling, some works resort to explore feature interactions in the DL-based recommendation models. LR [Cheng *et al.*, 2016], FM [Guo *et al.*, 2017], xDeepFM [Lian *et al.*, 2018] are exploited to import explicit low-order or high-order interactions through shortcut connections, i.e., concatenating interactions to the last hidden layer of MLP, and then apply linear projection with learnable variables or all-one vectors to generate the prediction scores jointly with the hidden layers. Other than the shortcut connection, FibiNet [Huang *et al.*, 2019] take interactions as the input of MLP, while PNN [Qu *et al.*, 2018] further combines interactions with raw embeddings to induce more features in the network. It is not hard to find that all these methods can be derived from AMEIR's search space. Concretely, all low-order and high-order interactions can be extracted from the search subspace of interactions. Meanwhile, raw input embeddings and explored interactions are used by both shortcut connection and MLP, covering the existing interaction-combining strategies. Besides, the attentive SE connection could implicitly characterize attention-based, regression-based, and FM-based shortcut connections.

MLP Investigation. The existing literature [Covington *et al.*, 2016; Zhou *et al.*, 2018] mainly uses sub-optimal handcrafted MLPs to aggregate features, where activation functions and hidden sizes are designed manually or by grid search. Different from these methods, AMEIR tries to find the appropriate MLP through one-shot NAS in a discrete yet flexible search subspace that covers common MLP settings.

Based on the above analysis, it can be inferred that AMEIR's search space covers most of the representative recommendation models. Therefore, AMEIR can perform as good as these methods in various scenarios, and further derive better models when conducting exhaustive search.

Model	Criteo		Avazu	
	AUC	Log Loss	AUC	Log Loss
DNN [Covington *et al.*, 2016]	0.7983	0.4549	0.7746	0.3831
Wide & Deep [Cheng *et al.*, 2016]	0.7992	0.4538	0.7749	0.3839
DeepFM [Guo *et al.*, 2017]	0.801	0.4496	0.7753	0.3825
IPNN [Qu *et al.*, 2018]	0.7975	0.4578	0.7787	0.3806
xDeepFM [Lian *et al.*, 2018]	0.8019	0.4497	0.7759	0.3818
FibiNet [Huang *et al.*, 2019]	0.8021	0.4487	0.7789	0.3798
AutoCross [Luo *et al.*, 2019]	0.8022 ± 0.0001	0.4494 ± 0.0001	0.7776 ± 0.0002	0.3810 ± 0.0001
AutoFIS [Liu *et al.*, 2020]	0.8015 ± 0.0004	0.4504 ± 0.0005	0.7789 ± 0.0003	0.3801 ± 0.0002
AutoCTR [Song *et al.*, 2020]	0.8027 ± 0.0003	0.4489 ± 0.0003	0.7791 ± 0.0002	0.3800 ± 0.0001
AMEIR-I	0.8024 ± 0.0002	0.4491 ± 0.0004	0.7791 ± 0.0003	0.3799 ± 0.0003
AMEIR-B	0.8030 ± 0.0002	0.4486 ± 0.0004	0.7798 ± 0.0004	0.3794 ± 0.0002
AMEIR-C	**0.8034** ± 0.0002	**0.4482** ± 0.0003	**0.7802** ± 0.0004	**0.3792** ± 0.0002

Table 3: Comparison with representative methods on Criteo/Avazu.

3 Experiments

In Sec.3, we will investigate AMEIR on various scenarios, perform additional ablation study and efficiency analysis.

3.1 Industrial Results for AMEIR

To verify all three steps of AMEIR in the industrial scenario, we refer to the Alimama CTR dataset [Feng *et al.*, 2019] that comprises both sequential and non-sequential features for a comprehensive comparison. We use similar preprocessing settings as [Feng *et al.*, 2019], where the embedding size is set to 4 and the maximum sequence length is set to 50. The Adam optimizer with an initial learning rate 1e-5 and batch size 1024 is employed for both AMEIR and baseline methods. For the training of the one-shot model, a single cosine schedule is introduced for learning rate decay. In Steps 1 and 3, AMEIR selects the top-5 models from the randomly sampled 2000 architectures for further evaluation, while in Step 2, AMEIR conducts 4 cycles of evolution, retains top-50 interactions at each cycle, and finally reserves $M = 5$ interactions as $\mathbf{p}^b$. L_a and L_c are set to 3 and 2 in the backbone model, and the pre-defined MLP is set to [200, 80] during Steps 1 and 2. By conforming to [Li and Talwalkar, 2019; Liu *et al.*, 2019], we run AMEIR for 4 times and report the mean metrics across 4 runs as the final results.

Tab. 1 depicts the results on Alimama dataset. AMEIR-A (only uses behavior modeling result from Step 1) presents competitive performance with the promising empirical architectures DIEN and DSIN. Furthermore, a significant improvement (note that an improvement of AUC around 0.0005-0.001 is already regarded as practically significant in the industrial scenarios [Feng *et al.*, 2019]) can be observed when combining AMEIR-A with searched interactions (AMEIR-B) and MLP (AMEIR-C). An interesting point on the Alimama dataset is that compared to sequential modeling and interaction exploration, the MLP search significantly contributes to the final results, which indicates that all three stages/steps in AMEIR are necessary for high-quality recommendations.

Model	AUC	Search Cost (GPU days)
Random	0.6321 ± 0.0011	2.9
RL [Zoph and Le, 2017]	0.6323 ± 0.0006	3.2
EA [Real *et al.*, 2019]	0.6338 ± 0.0004	3
ENAS (One-shot + RL) [Pham *et al.*, 2018]	0.6339 ± 0.0007	0.15
DARTS (One-shot + Gradient) [Liu *et al.*, 2019]	0.6350 ± 0.0008	0.2
SNAS (One-shot + Gradient) [Xie *et al.*, 2019]	0.6343 ± 0.0003	0.3
One-shot Random Search [Li and Talwalkar, 2019]	0.6365 ± 0.0006	0.3
AMEIR (Three-stage One-shot Random Search)	**0.6383** ± 0.0004	0.5

Table 4: Comparison with NAS methods on Alimama.

Model		Criteo		Avazu	
Interaction	MLP	AUC	Log Loss	AUC	Log Loss
None	Manual	0.7983	0.4549	0.7746	0.3831
MLP	Manual	0.8021	0.4492	0.7789	0.3801
FM	Manual	0.8024	0.4491	0.7791	0.3799
MLP + FM	Manual	0.8027	0.4488	0.7795	0.3795
MLP + SE	Manual	0.8030	0.4486	0.7798	0.3794
MLP + SE	Searched	**0.8034**	**0.4482**	**0.7802**	**0.3792**

Table 5: Comparison of backbone models on Criteo/Avazu.

3.2 Applicable for Various Scenarios

To show AMEIR is applicable for various recommendation tasks, we evaluate AMEIR on two additional scenarios, namely sequential scenario, and non-sequential scenario: (1) The sequential scenario only presents sequential user behaviors to examine Step 1 behavior modeling. The training is based on the item retrieval task. The sequential datasets include Amazon Beauty and Steam [Kang and McAuley, 2018]. (2) The non-sequential scenario is introduced to verify the Step 2 interaction exploration and Step 3 MLP investigation, where only non-sequential features are provided for the recommendation. The non-sequential datasets include benchmarking CTR datasets Criteo [Guo *et al.*, 2017] and Avazu [Qu *et al.*, 2018].

Evaluation on Sequential Scenario. Tab. 2 summarizes the results on Beauty and Steam datasets. The hand-crafted methods, including the strongest baselines BERT4Rec and NextItNet, can not dominate both datasets, which empirically justifies that the best architecture is usually task-aware. In contrast, AMEIR consistently achieves better accuracy compared to the baselines with well-designed architectures, indicating the strength of AMEIR and the capability of searching better models from the search subspace of building blocks.

Evaluation on Non-sequential Scenario. Tab. 3 presents the results on non-sequential datasets. It can be observed that AMEIR-I (DeepFM backbone with interactions explored by AMEIR) consistently surpasses competitive hand-crafted baselines xDeepFM and FibiNet as well as AutoML-based baselines AutoCross and AutoFIS on different tasks over multiple runs, validating the efficacy of interaction exploration in Step 2. Moreover, our method gains remarkable improvement over AMEIR-I and exceeds AutoCTR when combining explored interactions with the SE connection (AMEIR-B) and MLP investigation (AMEIR-C). The result verifies the design of both the backbone and search subspaces.

3.3 Ablation Study

Ablation Study on NAS Methods. To study the importance of the proposed search method, we evaluate the performance of different NAS algorithms on the same search space of the Alimama dataset. As shown in Tab. 4, the one-shot random search achieves better performance than both brute-force methods of random, RL, EA, and the algorithmic complex methods of ENAS, DARTS, SNAS with lower search cost. Moreover, AMEIR with the three-stage one-shot searching pipeline brings about a remarkable improvement on the one-shot random search under a similar time budget, demonstrating the effectiveness and efficiency of the proposed method.

Ablation Study on Backbone Model. To further analyze the impact of the backbone model, we isolate different components in Stages 2 and 3, and report the mean metrics on the non-sequential datasets. As shown in Tab. 5, the interactions (searched) applied to MLP and FM bring orthogonal improvement on vanilla DNN. The result reveals that both combining strategies contribute to the result. Additionally, SE interaction (AMEIR-B) and MLP search (AMEIR-C) could further boost the performance of recommendation, suggesting the superiority of the backbone design in AMEIR.

3.4 Efficiency in Search and Inference Phases

Search Efficiency. The search cost of AMEIR on the Beauty, Steam, Avazu, Criteo, Alimama datasets are 0.2, 0.2, 0.25, 0.25, 0.5 GPU-days respectively on a single Tesla V100 GPU, which are all similar to training the individual models of baseline methods. As AMEIR uses the trained one-shot weights to initialize the derived model, the fine-tuning time could generally be ignored. Therefore, the overall time cost of AMEIR is roughly the same as the methods that do not employ NAS. The result proves the efficiency of AMEIR in the search phase, and it is evident that AMEIR can help facilitate the design of recommender systems in real-world scenarios.

Inference Efficiency. We record the queries/samples-per-second (QPS) of AMEIR and the compared models to examine the inference speed on the Alimama dataset in Tab. 4. Apart from the improvement in accuracy, AMEIR is also competitive in efficiency compared with the commonly-used attention-based models. Besides, the searched MLPs are always much smaller than the hand-crafted ones, such that the numbers of parameters of the searched models are less than half of the baselines. The results indicate that instead of increasing the capacity, AMEIR coordinates the three parts in the recommendation model to find the ideal architecture.

4 Conclusion

In this paper, we introduce AMEIR for automatic behavior modeling, interaction exploration, and MLP investigation in the recommender systems. Owing to the design of novel search space and search pipeline, AMEIR outperforms the competitive baselines in various scenarios with lower model complexity and comparable time cost, demonstrating efficiency, effectiveness, and robustness of the proposed method.

Acknowledgements

This work is partially supported by National Key Research and Development Program No. 2017YFB0803302, Beijing Academy of Artificial Intelligence (BAAI), NSFC 62032003 and 61632017.

References

[Ba *et al.*, 2016] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[Cheng *et al.*, 2016] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *DLRS*, pages 7–10, 2016.

[Covington *et al.*, 2016] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *RecSys*, 2016.

[Feng *et al.*, 2019] Yufei Feng, Fuyu Lv, Weichen Shen, Menghan Wang, Fei Sun, Yu Zhu, and Keping Yang. Deep session interest network for click-through rate prediction. In *AAAI*, pages 2301–2307. AAAI Press, 2019.

[Guo *et al.*, 2017] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: a factorization-machine based neural network for ctr prediction. In *IJCAI*, pages 1725–1731, 2017.

[Hidasi and Karatzoglou, 2018] Balázs Hidasi and Alexandros Karatzoglou. Recurrent neural networks with top-k gains for session-based recommendations. In *CIKM*, pages 843–852. ACM, 2018.

[Hu *et al.*, 2018] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, 2018.

[Huang *et al.*, 2019] Tongwen Huang, Zhiqi Zhang, and Junlin Zhang. Fibinet: combining feature importance and bilinear feature interaction for click-through rate prediction. In *RecSys*, pages 169–177, 2019.

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

[Li and Talwalkar, 2019] Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. In *UAI*, 2019.

[Li *et al.*, 2017] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. Neural attentive session-based recommendation. In *CIKM*, 2017.

[Lian *et al.*, 2018] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *KDD*, 2018.

[Liu *et al.*, 2018] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *ECCV*, 2018.

[Liu *et al.*, 2019] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In *ICLR*, 2019.

[Liu *et al.*, 2020] Bin Liu, Chenxu Zhu, Guilin Li, Weinan Zhang, Jincai Lai, Ruiming Tang, Xiuqiang He, Zhenguo Li, and Yong Yu. Autofis: Automatic feature interaction selection in factorization models for click-through rate prediction. In *KDD*, 2020.

[Luo *et al.*, 2019] Yuanfei Luo, Mengshuo Wang, Hao Zhou, Quanming Yao, Wei-Wei Tu, Yuqiang Chen, Wenyuan Dai, and Qiang Yang. Autocross: Automatic feature crossing for tabular data in real-world applications. In *KDD*, pages 1936–1945, 2019.

[Pham *et al.*, 2018] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In *ICML*, 2018.

[Qu *et al.*, 2018] Yanru Qu, Bohui Fang, Weinan Zhang, Ruiming Tang, Minzhe Niu, Huifeng Guo, Yong Yu, and Xiuqiang He. Product-based neural networks for user response prediction over multi-field categorical data. *TOIS*, 2018.

[Ramachandran *et al.*, 2017] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

[Real *et al.*, 2019] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *AAAI*, 2019.

[Rendle *et al.*, 2009] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *UAI*, pages 452–461. AUAI Press, 2009.

[Song *et al.*, 2020] Qingquan Song, Dehua Cheng, Hanning Zhou, Jiyan Yang, Yuandong Tian, and Xia Hu. Towards automated neural interaction discovery for click-through rate prediction. In *KDD*, pages 945–955, 2020.

[Sun *et al.*, 2019] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *CIKM*, 2019.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, pages 5998–6008, 2017.

[Xie *et al.*, 2019] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. Snas: stochastic neural architecture search. In *ICLR*, 2019.

[Yuan *et al.*, 2019] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M Jose, and Xiangnan He. A simple convolutional generative network for next item recommendation. In *WSDM*, pages 582–590, 2019.

[Zhou *et al.*, 2018] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. Deep interest network for click-through rate prediction. In *KDD*, pages 1059–1068, 2018.

[Zhou *et al.*, 2019] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. Deep interest evolution network for click-through rate prediction. In *AAAI*, pages 5941–5948, 2019.

[Zoph and Le, 2017] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. In *ICLR*, 2017.