

AdaS&S: a One-Shot Supernet Approach for Automatic Embedding Size Search in Deep Recommender System

He Wei*

Machine learning platform
department, Tencent TEG.
Beijing, China
whywei@tencent.com

Yuekui Yang*†

Machine learning platform
department, Tencent TEG. And,
Department of Computer Science and
Technology, Tsinghua University.
Beijing, China
yuekuiyang@tencent.com

Yang Zhang

Machine learning platform
department, Tencent TEG.
Beijing, China
yizhizhang@tencent.com

Haiyang Wu

Machine learning platform
department, Tencent TEG.
Beijing, China
gavinwu@tencent.com

Meixi Liu

Machine learning platform
department, Tencent TEG.
Beijing, China
meixiliu@tencent.com

Shaoping Ma

Department of Computer Science and
Technology, Tsinghua University.
Beijing, China
msp@tsinghua.edu.cn

ABSTRACT

Deep Learning Recommendation Model(DLRM)s utilize the embedding layer to represent various categorical features. Traditional DLRMs adopt unified embedding size for all features, leading to suboptimal performance and redundant parameters. Thus, lots of Automatic Embedding size Search (AES) works focus on obtaining mixed embedding sizes with strong model performance. However, previous AES works can hardly address several challenges together: (1) The search results of embedding sizes are unstable; (2) Recommendation effect with AES results is unsatisfactory; (3) Memory cost of embeddings is uncontrollable. To address these challenges, we propose a novel one-shot AES framework called AdaS&S, in which a supernet encompassing various candidate embeddings is built and AES is performed as searching network architectures within it. Our framework contains two main stages: In the first stage, we decouple training parameters from searching embedding sizes, and propose the Adaptive Sampling method to yield a well-trained supernet, which further helps to produce stable AES results. In the second stage, to obtain embedding sizes that benefits the model effect, we design a reinforcement learning search process which utilizes the supernet trained previously. Meanwhile, to adapt searching to specific resource constraint, we introduce the resource competition penalty to balance the model effectiveness and memory cost of embeddings. We conduct extensive experiments on public datasets to show the superiority of AdaS&S. Our method could improve AUC by about 0.3% while saving about 20% of model

*Both authors contributed equally to this research.

†Corresponding author.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

XXX '23, XX XXXX, 2023, XXXX, XXX, XXX

© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN XXX XXX...\$15.00

<https://doi.org/XXXXXXXXXXXXXX>

parameters. Empirical analysis also shows that the stability of searching results in AdaS&S significantly exceeds other methods.

CCS CONCEPTS

• Information systems → Recommender systems.

KEYWORDS

Recommendation, Embedding size, Supernet, Reinforcement Learning

ACM Reference Format:

He Wei, Yuekui Yang, Yang Zhang, Haiyang Wu, Meixi Liu, and Shaoping Ma. 2023. AdaS&S: a One-Shot Supernet Approach for Automatic Embedding Size Search in Deep Recommender System . In *Proceedings of XXXXXX XXXXXX, XXX XXXX, 2023, XXXX, XXXX, XXX*. ACM, New York, NY, USA, 10 pages. <https://doi.org/XXXXXXXXXXXX>

1 INTRODUCTION

Deep Learning Recommender Model(DLRM)s often take a large amount of categorical features as input, and utilize a embedding layer to convert them to low-dimensional embedding vectors [1]. Consequently, the embedding layer plays an important role since it dominates the number of parameters as well as the effect of model prediction [2, 3]. Traditional DLRMs generally assign a unified embedding size (abbreviated as “Emb-size” in this work). However, unified Emb-size ignores the heterogeneity of features and suffers from following issues: (1) Inferior performance: setting a unified Emb-size could impair the expressing of feature fields with large cardinality (number of unique feature values), while causing over-fitting for those with low cardinality. (2) Memory inefficiency: Unified Emb-size could cause redundant embedding parameters as well as unnecessary computation overhead, especially for features contribute less to final prediction. Therefore, different Emb-sizes is highly desired now and become a focal point in researches, especially for online recommendation where inference speed and memory cost become the bottleneck.

We denote the task of obtaining different Emb-size for features as AES (Automatic Embedding size Search). Existing AES methods

can be categorized as heuristic, pruning and NAS-based methods. Heuristic methods tend to allocate Emb-sizes with pre-defined rules. For example, MDE [4] utilizes the popularity of features to determine the Emb-sizes, but its capability is limited by the degeneration of model performance [3]. As an alternative to heuristic methods, pruning methods [3, 5–7] reduce Emb-sizes from a pre-defined largest dimension according to learnable threshold or mask parameters. For example, AutoSrh [7] use soft selection weights to identify the importance of each dimension. However, the interaction of thresholds (masks) and embeddings undermines the stability of their searching outcomes, and the complex relation between dimensions is not well-studied. Recent NAS-based approaches [2, 8–10] view AES as searching candidate network architectures, they utilize NAS techniques to achieve better performance. Typically, AutoDim [2] introduces additional weights for embeddings of different Emb-sizes into the network, and optimize them with DARTS [11]. However, parameter training and dimension searching in these works are usually coupled, leading to several weaknesses: (1) joint optimization of training and searching could introduce bias during gradient descent and mislead the AES to unstable results. (2) the number of embedding parameters is not guaranteed, raising the risk of resource costs for model deployment. To sum up, though recent pruning and NAS-based methods outperform traditional methods in model effectiveness, they still suffer from the instability of searched results, and it is also challenging to adapt to varying resource constraint (i.e., memory cost of embedding parameters) while seeking for better Emb-sizes.

In this paper, we propose a novel approach called AdaS&S (**A**daptive **S**upernet and **R**L-**S**earch) to address the aforementioned challenges. A large network containing all candidate embeddings is built (known as “supernet”), and Emb-sizes are searched as “subnet” architecture, i.e., choosing particular embedding tables. To be specific, (1) *How to generate stable results?* The decoupled training and searching eliminates the bias in joint optimization [12], helps to obtain stable outcomes. Meanwhile, a supernet with robust parameters could alleviate the uncertainty in searching. (2) *How to obtain better performance?* We propose the Adaptive Sampling to improve the consistency of the trained supernet, so that subnets inheriting weights from it could be quite predictive for stand-alone trained ones. A Reinforcement Learning (RL) based searching strategy is further designed to determine the optimal allocation. (3) *How to adapt Emb-sizes to resource constraint during searching?* We utilize a regularization of total resource cost to guide the RL agent, so that it produces controllable Emb-sizes.

Our approach is experimented on several real-world datasets and shows significant advantages. Besides, the searching can repeat many times given various resource constraints once the supernet is trained, which is especially suitable for the dynamic environment of online recommendation. In summary, the main contribution of this paper are as below:

- We firstly summarize three main goals for an ideal AES method: stable searching results, better recommendation performance with searched Emb-sizes, and being adaptive to specific resource constraint. We propose the AdaS&S framework to systematically fulfill all these targets.
- Our AdaS&S method models AES by a one-shot two-stage supernet, which successfully decouples Emb-size searching from

embedding training. Adaptive Sampling is proposed to perform the supernet training. Also, we design a Reinforcement Learning method to obtain the (near-)optimal Emb-size in the searching space. A regularization for total resource is introduced to guide the search result under resource constraints.

- We conduct extensive experiments on several datasets to demonstrate the superiority of our approach over other competitive methods. Empirical results show that AdaS&S could successfully address mentioned challenges of AES.

2 PRELIMINARY

2.1 Features in DLRMs

In real-world large-scale recommendation system [13–17], there are a lot of different feature fields, in which categorical features account for a large portion. Those features are often stored in high-dimensional one-hot or multi-hot vector, and are further converted to meaningfully embeddings in the latent space. Characteristics of these feature fields can be quite different. First, the number of distinct feature values (i.e. cardinality) in each fields varies a lot. Second, there may be semantic overlapping and value correlation among feature fields. Last, some feature fields are much more important than others in predicting recommendation results. Therefore, a non-unified assignment of Emb-sizes is believed to improve the effect and efficiency of DLRM.

2.2 Base Architecture of DLRM

DLRMs can be implemented as various popular architectures like WDL [14], DeepFM [15], DIN [17]. There are usually three base components in those models: embedding layer, main model, and the output layer, which is shown in Figure 1.

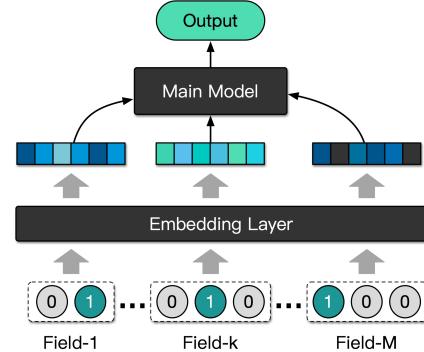


Figure 1: Base architecture of the DLRMs.

Embedding Layer: Assume that there are M categorical feature fields in input data, each feature is represented as a binary one-hot or multi-hot encoded vector as mentioned above, then the data sample can be formulated as $\mathbf{x} = [x_1, x_2, \dots, x_M]$, where x_i is one-hot or multi-hot vector of the i -th field. The embedding layer then map these x_i to low-dimensional real-valued vectors $e_i = V_i \cdot x_i$, where $V_i \in R^{n_i \times d}$ is the embedding space of i -th field, d is the embedding dimension, n_i is the cardinality of the i -th field. All parameters in embeddings (a.k.a. embedding table) is $V = \{V_1, V_2, \dots, V_m\}$. The output of embedding layer is $E = (e_1, e_2, \dots, e_m)$.

Main Model: The main model takes E as input, and employs some advanced network architectures like Multi-Layer Perception(MLP)

[18], Factorization Machine(FM) [19] and Transformer [20] to further process the input information. Denoting the parameters of main model as Θ , then the mapping f from input x to prediction \hat{y} can be formulated as:

$$\hat{y} = f(x|V, \Theta), \quad (1)$$

Output Layer: Typically in CTR prediction, the problem is formulated as a binary classification task, the output \hat{y} of DLRM is a predict score for the likelihood of click. During training, the cross-entropy loss is utilized for updating the network:

$$\mathcal{L}(y, \hat{y}; V, \Theta) = -y \log(\hat{y}) - (1-y) \log(1-\hat{y}), \quad (2)$$

where $y \in \{0, 1\}$ is the ground-truth label.

2.3 One-Shot Supernet

Recent NAS approaches adopt weight sharing to avoid the expensive computation. All candidate network architectures (i.e., subnet) inherit weights from a large network (i.e., supernet). The design of decoupling the supernet training from the architecture search is known as One-Shot NAS [21–25], which often consists of following four steps: (1) define a search space and implement it as a supernet. (2) train the supernet weights to make subnets similar to those trained stand-alone. (3) search candidate architectures using the pre-trained supernet with various strategies. (4) select the top performing architecture and retrain it.

The key difference of one-shot supernet and DARTS [11] is that one-shot supernet conducts step (2) and (3) in a sequential manner [12]. Since supernet trained in Step (2) guides the searching process in Step (3), the evaluation results using inherited supernet weights should be predictive for those using stand-alone trained weights. To evaluate the quality of trained supernet, ranking correlation metric, e.g. Kendall Tau, is employed to measure the consistency between one-shot models and stand-alone trained ones [26].

3 FRAMEWORK

The main goal of AES in this work is to select the optimal Emb-sizes for each feature field to achieve best recommendation performance. As in previous works like AutoDim [2] and ESAPN [9], we can formulate this into an optimization problem as follows:

$$\mathbf{d}^*, \mathbf{V}^*, \Theta^* = \arg \min_{\mathbf{d} \in \mathcal{D}} \mathcal{L}(y, \hat{y}; \mathbf{d}, \mathbf{V}, \Theta), \quad (3)$$

where $\mathbf{d} = [d_1, d_2, \dots, d_M]$ is the Emb-size assignment result for M features, i.e., $V_i \in R^{n_i \times d_i}$. \mathcal{D} is the search space of the Emb-size, if there are M features and T candidate Emb-sizes, the search space \mathcal{D} will contain T^M elements. As M and T can be quite large, it is impossible to traverse all cases by Eq.(3) to obtain the optimal Emb-size assignment due to the computation cost.

To solve Eq.(3), we propose the two-stage one-shot framework AdaS&S. It contains two main stages: training supernet and searching subnet (see Figure 2). To be more precise, (1) in the training stage, we train the supernet that encompasses all candidate Emb-sizes and aim to make all embeddings fully expressive. We design the Adaptive Sampling method to obtain a supernet with informative parameters and high consistency. (2) in the searching stage, we search the optimal Emb-size with a RL process. A policy network serves as RL agent that takes the state (chosen Emb-sizes

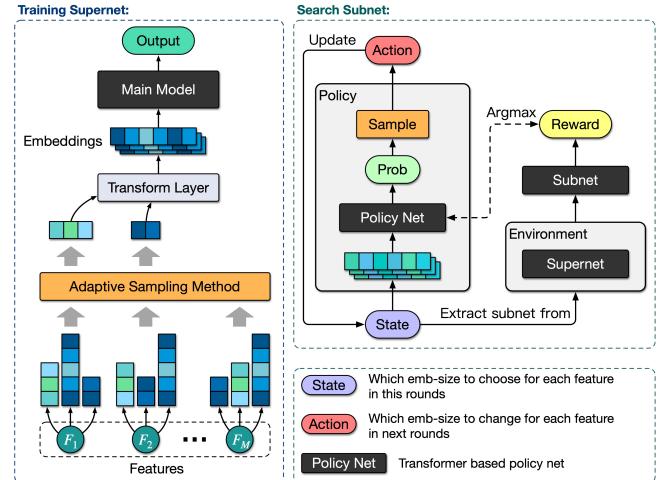


Figure 2: Overview of the proposed AdaS&S framework. The left part is the stage of training the Adaptive Supernet. The right part is the stage of RL-Search process.

at current step) and yields the action (chosen Emb-sizes for next step) under the resource competition penalty. The RL agent keeps adjusting Emb-sizes until this process ends (the searching being converged); (3) finally, an additional retraining step is conducted to produce a DLRM with different Emb-sizes where only optimal Emb-sizes are taken into consideration.

Table 1 gives a comprehensive comparison of our approach with others on whether they addressed aforementioned challenges. To the best of our knowledge, AdaS&S is the only method that tackles all challenges in design. Moreover, our framework could be implemented into various DLRMs, and the one-shot nature (supernet training decoupled with Emb-size searching) allows multiple times of searching after the training stage, which is particularly efficient for online applications with ever-changing requirements. Next we will present details of several main components in AdaS&S.

Table 1: Capabilities of AES methods. “Stable” for obtaining stable search results, “Performance” for improving model performance with new Emb-sizes, “Res” for adapting to the resource constraint of parameters.

Model	Category	Stable	Performance	Res
MDE [4]	Heuristic	✓	✗	✓
AutoDim [2]	NAS	✗	Sub ¹	✗
AutoEmb [8]	NAS	✗	Sub	✗
ESAPN [9]	NAS	✗	Sub	✗
AMTL [3]	Pruning	✗	Sub	✗
PEP [5]	Pruning	✗	Sub	✗
SSEDS [6]	Pruning	✗	Sub	✓
AdaS&S	NAS (One-Shot)	✓	Better ²	✓

3.1 Adaptive Supernet

Following the paradigm in Section.2.3, we propose our one-shot supernet to encompass all possible subnets (Emb-size allocations) in the search space. This enables our method to have the following strengths: (1) parameters in embeddings and main model could

¹“Sub” indicates sub-optimal performance (than AdaS&S).

²“Better” indicates better performance than others

be decoupled from searching Emb-sizes; (2) searching can be conducted independently and repeatedly after supernet training. Our supernet shall also meet the following requirements: (1) it should be memory-efficient when handling with larger search space \mathcal{D} , (2) the training of parameters shall be adaptively adjusted in order to fully express each candidate embeddings, (3) the training stage shall produce a supernet with high consistency to better guide the searching, and (4) training supernet shall be easily integrated into various DLRMs. To meet these requirements, we introduce several techniques into our supernet training, including supernet schemes, Adaptive Sampling, and unifying different Emb-sizes.

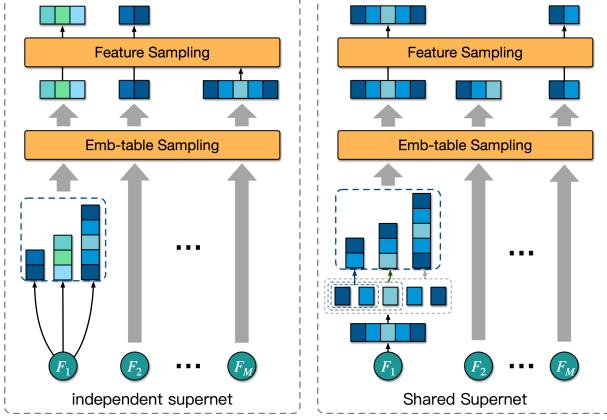


Figure 3: Two schemes of supernet training.

Supernet Scheme: We devise two different supernet schemes to fulfill candidate embeddings with different sizes, namely independent supernet and shared supernet (Figure 3). In *Independent Scheme*, we initialize several independent embedding tables for each candidate Emb-sizes, each individual table serves for one candidate Emb-size. This scheme ensures each candidate trained independently, producing more accurate parameters. Yet it may place greater demands on hardware during training. The number of total embedding parameters in this scheme is $\sum n_i \sum_{t=1}^T d_t$.

While larger search space \mathcal{D} is preferred and memory cost of supernet becomes a concern, *Shared Scheme* is a feasible alternative, in which a largest embedding table (known as “max-table”) with largest candidate Emb-size is initialized and smaller candidates with Emb-size d_t are parts of it (by selecting first d_t dimensions). This scheme is more memory-efficient especially for larger M, T .

Adaptive Sampling: Sampling candidate architectures plays a key role in supernet training and shows a significant impact on supernet consistency [12, 21]. To determine a proper way to train the supernet, we devise our new sampling method from two perspectives: On the one hand, to obtain well-trained parameters in different Emb-sizes, it is preferable to increase the chance for training under-fitting embedding tables. On the other hand, to distinguish the contribution of different features fields, features that are more important should fully express their information, thus requiring higher training intensity. Consequently, we propose a two-layer Adaptive Sampling method:

$$\tilde{\mathbf{E}} = AdaS(\mathbf{E}, P_E, P_F) = FS(\dot{\mathbf{E}}, P_F), \quad (4)$$

Where, $\dot{\mathbf{E}} = ES(\mathbf{E}, P_E)$.

In Eq.(4), $\mathbf{E} = \{E_{i,j}\}_{M \times T}$ is the embedding list of M features and T candidate Emb-sizes, where $E_{i,j} \in R^{d_j}$, $P_E \in R^{M \times T}$ is the sample rate of Emb-sizes, $P_F \in R^M$ is the sample rate of features. The two layers of Adaptive Sampling consist of:

- $ES(\mathbf{E}, P_E)$ is the first layer called embedding table sampling. It builds a categorical distribution over embedding list \mathbf{E} according to sample rate P_E . Each time it samples a result $\dot{\mathbf{E}} = \{E_i\}_M$, which is the chosen embeddings for M features. For each feature, ES selects exactly one embedding from T candidates.

- $FS(\dot{\mathbf{E}}, P_F)$ is the second layer called feature sampling. It builds a multivariate Bernoulli distribution over $\dot{\mathbf{E}}$ according to sample rate P_F . Each time it samples a result $\tilde{\mathbf{E}} = \{E_i\}_{\tilde{M}}$, where $\tilde{M} \leq M$. $\tilde{\mathbf{E}}$ is the final embedding list of selected \tilde{M} features. In other words, FS would select a subset of all M features to train.

Similar to WAST [27], our sample rate P_E is adaptively updated based on the training status of embeddings, and P_F is adaptively updated based on the importance of features. We formulate the updating of P_E and P_F as below:

$$P_E^{(t+1)} = P_E^{(t)} - LN\left(\frac{1}{T} \sum_{j=1}^T Var(E_{.,j})\right), \quad (5)$$

$$P_F^{(t+1)} = P_F^{(t)} + LN\left(\frac{\lambda_{FS}}{M \cdot T} \|\frac{\partial \mathcal{L}}{\partial \dot{\mathbf{E}}} \|_1 + \frac{1 - \lambda_{FS}}{M \cdot T} \|\dot{\mathbf{E}}\|_1\right), \quad (6)$$

LN is the layer normalization, $Var(E_{.,j})$ is the value variance of the j -th candidate embedding in a field, $\partial \mathcal{L} / \partial \dot{\mathbf{E}}$ is the gradient for embeddings, $\|\dot{\mathbf{E}}\|_1$ is the L1 norm of embeddings, and λ_{FS} is the hyperparameter for balancing the effects of gradients and L1 norms. By Eq.(5), embedding table with insufficiently trained parameters (lower variance) will get more chance to be sampled than others. By Eq.(6), features contribute more to prediction (larger gradients or L1 norm) will get more chance to be sampled.

Unifying Different Emb-sizes: After sampling, we obtain the embedding list $\tilde{\mathbf{E}}$ containing mixed Emb-sizes. Since some DLRMs like DeepFM [15] require same dimensions for input embeddings from all feature fields to model feature interactions, we need to transform embeddings in $\tilde{\mathbf{E}}$ into same Emb-size in order to feed them into main model:

$$\tilde{\mathbf{E}} = Transform(\tilde{\mathbf{E}}, d_f), \quad (7)$$

where $\tilde{\mathbf{E}} \in R_{M \times d_f}$ is the final embeddings of M features, d_f is the unified size. $Transform$ is implemented as MLP layers, and we introduce T MLPs to map T candidate Emb-size d_1, \dots, d_T to d_f respectively. All candidate embeddings with same Emb-size across feature fields shall be transformed with the same MLP, which helps to reduce parameters and computation overhead during training stage. Batch Normalization [28] is appended to MLPs to stabilize the magnitude of transformed embeddings according to [2, 8]. The overall training process of supernet is shown in Algorithm.1.

3.2 RL-Search Process

After the training stage, we can search the optimal subnet based on the parameters of supernet. In NAS, genetic algorithm, evolution algorithm, RL, etc. [29] are commonly used for searching the architecture, and we employ RL for AdaS&S in this stage. To be specific, we design a transformer based policy net capturing the complex

Algorithm 1: Supernet training process

```

Input: Features:  $x = [x_1, \dots, x_M]$ , Labels:  $y$ , Supernet
       scheme:  $S$ , Candidate Emb-sizes:  $D = [d_1, d_2, \dots, d_T]$ ,
       Final Emb-size:  $d_f$ ;
Output: the well-trained Supernet;
1 Initialize  $P_E, P_F$ ;
2 if  $S = \text{Independent}$  then
3   Initialize several embedding tables of  $D$ :  $E = \{E_{i,j}\}_{M \times T}$ ;
4 else if  $S = \text{Shared}$  then
5   Initialize the maximum table:  $E_{\max}$ ;
6   Intercept  $E_{\max}$  into several tables of  $D$ :  $E = \{E_{i,j}\}_{M \times T}$ ;
7 for  $batch = 0$ ;  $batch < B$ ;  $batch++$  do
8   Feed a mini-batch:  $\{x, y\}$  of training set to Supernet;
9   Embedding lookup from  $E$  based on  $x$ ;
10  Adaptive Sampling:  $\tilde{E} = \text{AdaS}(E, P_E, P_F)$ ;
11  Transform to unified size:  $\tilde{E} = \text{Transform}(\tilde{E}, d_f)$ ;
12  Update  $W$  by descending  $\nabla_W \mathcal{L}_{\text{train}}(\hat{y}(\tilde{E}), y)$ ;
13  Update Sample rate:  $P_E, P_F$  by Eq.(5) and Eq.(6);
14 end

```

relationships between features to conduct the search process. The components of our RL-search process can be formulated as:

- (1) **State:** $s \in R^M$, the state of Emb-size assignment.
- (2) **Action:** $a \in R^M$, select the Emb-size for each feature in the next round.
- (3) **Track:** $\tau_l = [s_0, a_0, s_1, a_1, \dots, s_l, a_l]$ is the state-action track in l -th step in RL process.
- (4) **Policy:** $\pi_\theta(s_i)$ is a transformer based policy network. It takes current **state** as inputs and generates a state transition probability matrix $\mathcal{P} \in R^{M \times T}$, where $\mathcal{P}_{i,j}$ is transition probability of the i -th feature from the current Emb-size to the j -th Emb-size. When RL-searching is converged, $\mathcal{P}_{i,j}$ will represent the probability that the j -th Emb-size will be selected for the i -th feature.
- (5) **Reward:** $\mathcal{R}(s_i, a_i) = \lambda_r \cdot ACC_{\text{val}}(\hat{y}, y)$, where λ_r is the weight of reward, $ACC_{\text{val}}(\hat{y}, y)$ is the accuracy of the subnet.
- (6) **Objective function:** We define the objective $\mathcal{J}_{\text{val}}(a_i|\theta)$ based on REINFORCE [30] and AutoFSS [31], it can be expanded as:

$$\mathcal{J}_{\text{val}}(s_i, a_i|\theta) = -\mathbb{E}_{\pi_\theta(s_i)} [\log(p_\theta(\tau_i)) \cdot \mathcal{R}(s_{i-1}, a_i)] + \mathbb{P}, \quad (8)$$

where, $\mathbb{E}_{\pi_\theta(a_i)}[\cdot]$ denotes the expected reward of the track τ_i . \mathbb{P} is resource competition penalty, which is introduced to limit the total memory resources occupied by embedding tables.

The resource competition penalty \mathbb{P} can be formulated as:

$$\mathbb{P} = \lambda_r \frac{1}{M} \sum_{i=1}^M \sum_{j=1}^T (d_j \mathcal{P}_{i,j}) - \lambda_c \frac{1}{M} \sum_{i=1}^M \| \mathcal{P}_i - \frac{1}{T} \|_2, \quad (9)$$

where λ_r, λ_c is the hyper-parameter for weights of penalties, d_j is the j -th Emb-size in search space, T is the number of candidate Emb-sizes. The first part of \mathbb{P} is the Emb-size resources penalty, which can limit the memory resources. The second part is a competition penalty, which keeps Emb-size away from a moderate result, i.e., uniformly choosing a candidate size. In this way, RL searching can model the competency of features for resources and grant larger Emb-sizes for more promising features.

The RL-search process is illustrated in Figure 2 (right side). After searching, we can obtain the optimal Emb-size assignment according to transition probability \mathcal{P} . It is notable that by adjusting λ_r and λ_c in the penalty \mathbb{P} , we can balance the effect and resource during searching. For example, to force the searching to generate Emb-sizes with less embedding parameters, we can increase the value of λ_r .

3.3 Re-train with Different Emb-size

After the searching stage, optimal Emb-size assignment is obtained by \mathcal{P} . For the m -th feature, optimal Emb-size d_m^* is the t_m^* -th candidate Emb-size, and: $t_m^* = \text{argmax}_t(\mathcal{P}_m)$. Since the supernet is trained by sampling embeddings of different sizes, a re-training of main model with only optimal E_m^* is desired to eliminate the influence of suboptimal Emb-sizes. As in Section.2, we obtain unique embedding vectors $[e_1, e_2, \dots, e_M]$ for features $[x_1, x_2, \dots, x_M]$, and concatenate them to feed into hidden layers of the main model. As in Section.3.1, we still employ the transform layer to map embeddings of mixed-sizes to a unified size. Without redundant embedding parameters, the memory consumption and training time would considerably decrease in re-training.

4 EXPERIMENT

In this section, we design experiments to analyze the following research questions:

- **RQ1:** How does the proposed AdaS&S perform on AES problem? Could AdaS&S outperform other past AES methods for obtaining optimal recommendation performance?
- **RQ2:** Can the proposed AdaS&S obtain searching results with more stability?
- **RQ3:** Can the proposed AdaS&S adaptively search for Emb-sizes according to specified resource constraints?
- **RQ4:** Does Adaptive Sampling in the training stage improve the supernet consistency?
- **RQ5:** What is the influence of AdaS&S on the efficiency of searching Emb-sizes and model inference?

4.1 Dataset

We conduct experiments on three real-world datasets:

Avazu³ consists of 40M users' click records on ads over 11 days, and each record contains 22 categorical features. We chronologically divide this dataset into training/validation/test set by 8:1:1.

Movielens-1M (ML-1M)⁴, a dataset for about 1M records of user-movie ratings. As in AutoInt [32] and PEP [5], we take samples with ratings > 3 as positive samples. The original data samples contain several feature fields: user ID, gender, age, occupation, zip, movieID, year, genres, timestamps. We convert timestamp to 2 categorical features, namely weekday (indicating whether it is for weekend or not) and hour_in_day (indicating the time in 24 hour). The training/validation/test set is divided by 8:1:1 based on timestamp.

Criteo⁵, a dataset contains 13 numerical and 26 categorical features. Note that all feature fields are anonymized with MD5

³<https://www.kaggle.com/competitions/avazu-ctr-prediction/data>

⁴<https://grouplens.org/datasets/movielens/1m/>

⁵<https://www.kaggle.com/datasets/mrkmakr/criteo-dataset>

encoding. We normalize numerical features by transforming a value v as in [2], and then bucket them into categorical features. We divide the data samples into training/validation/test set with 8:1:1 according to the history of user-item interactions. The summary of these datasets can be found in Table 2.

Table 2: Overall information of datasets.

Data	Avazu	ML-1M	Criteo
# Interactions	40,349,512	994,833	45,840,617
# Feature Fields	22	10	39
# Feature Values	6,481,134	15,703	7,605,436
Label	click or not	Rating	click or not

4.2 Implement Details

Base Model: We compare AdaS&S with several popular AES methods: MDE [4], AutoDim [2], AutoEmb [8], ESAPN [9], SSEDS [6]. We also compare our method with the case where all embeddings are in unified Emb-size (dubbed as UES). For example, UES-32 means all Emb-sizes are set to 32. UES could set a baseline for model performance and parameter number. We conduct AES and UES settings based on popular DLRM architectures, namely Wide&Deep [14] and DeepFM [15]. Note that the original AutoEmb and ESAPN would search Emb-sizes for each unique feature value, which is hardware-unfriendly [33] for industrial-level applications. In our experiments, we employ the main framework of these two methods and conduct them in a field-wise manner. We employ the original MDE since it is a typical heuristic method which determines Emb-sizes by the popularity of feature values.

Hyper-parameter: The hyper-parameters in experiments are as below: **(a)** Batch size is 512 for three datasets. **(b)** Learning rate: 0.001 for Main Model (Supernet training stage), and 0.0005 for searching stage. **(c)** MLP layer: Wide&Deep and DeepFM use a 3-layer MLP network with ReLU as the activation function. The width of MLP layer is set to {128, 64, 1} for Avazu and ML-1M, and {256, 128, 1} for Criteo. **(d)** Search space: $D = \{2, 8, 16, 32, 64\}$. The largest candidate Emb-size is 64. **(e)** Settings of AdaS&S: We use independent supernet scheme for ML-1M, shared supernet scheme for Criteo and Avazu. In feature sampling process, $\lambda_{FS} = 0.6$. In resource competition penalty, $\lambda_r = 0.0025$ and $\lambda_c = 0.08$ for effect-first mode, $\lambda_r = 0.005$ and $\lambda_c = 0.04$ for resource-first mode.

Metric: Following DeepFM [15], we use the commonly-used metrics for CTR prediction: **AUC** (Area Under ROC), higher AUC means better performance. Note that 0.1% growth of AUC is considered as a significant improvement [15] in the recommendation system.

LogLoss (cross-entropy loss of the model), lower is better. To analyze memory costs of parameters in retrain stage, we also utilize **P-R**: the percentage of Parameter Reduction compared with UES-32.

4.3 Overall Performance (RQ1)

We now analyze the performance of various AES methods mentioned in Section 4.2. Note that we conduct AdaS&S in two different modes: **AdaS&S-R** is the resource-first mode, where larger penalty for resource constraint is applied; **AdaS&S-E** is the effect-first mode, where searching stage will seek Emb-sizes with best recommendation accuracy regardless of resource competition

penalty. The overall performance is shown in Table 3. We can observe that:

1. The impact of Emb-size is significant. It affects not only the effects of recommendation, but also the memory cost of model parameters. By comparing UES-32 and UES-24, reducing Emb-size from 32 to 24 will cause a decrease of $0.3\%\pm$ in AUC and an increase of $0.4\%\pm$ in LogLoss, with a parameter reduction $24\%\pm$.
2. From the perspective of saving memory, MDE reduces the parameters by $20\%\pm$, but it cannot guarantee the effect of recommendation. Under the premise of keeping the recommendation effect unchanged, AutoDim and AutoEmb reduces the parameters by $10\%\pm$ (unstable), ESAPN and SSEDS reduces the parameters by 20% . Excitingly, AdaS&S-R outperforms other AES Method with $30\%\pm$ parameters reduction and bring a little improvement of recommendation effect.
3. From the perspective of improving the recommendation effect, Past AES methods as well as AdaS&S-R can improve AUC by roughly $0.1\%\pm$ and decrease LogLoss by $0.15\%\pm$. It is impressive that AdaS&S-E can improve AUC by $0.3\%\pm$ and decrease LogLoss by $0.35\%\pm$, which considerably outperforms others.

To summarize, AdaS&S-E can successfully improve the recommendation effect by above 0.3% increase of AUC and reduce $20\%\pm$ of parameters. Meanwhile, AdaS&S-R can significantly reduce $30\%\pm$ of parameters and improve the recommendation effect by about 0.1% increase of AUC. AdaS&S significantly outperforms AES methods in terms of effect and stability.

4.4 Detailed Analysis

In this section, we further investigate characteristics of AdaS&S to address several RQs discussed in Section 4.

4.4.1 Robustness of Emb-size Searching (RQ2)

The decoupling of training and searching stage in AdaS&S, as well as the design of Adaptive Sampling, is believed to greatly improve the stability of produced search results. To analyze the stability of searching, we conduct a series of experiments to investigate the distribution of selected Emb-sizes for features using AdaS&S and other popular AES methods. For example, on ML-1M dataset we use different AES methods (AdaS&S, AutoDim, AutoEmb, ESAPN) and perform 100 times of searching respectively. For each feature, if the Emb-size assigned to it is more consistent across these 100 results, then the corresponding AES method is believed to be more robust. We analyze the Emb-size searching results for 10 feature fields in ML-1M, and the result is shown in Figure 4.

In Figure 4, for each feature AdaS&S produces the same Emb-size with a probability of more than 75%, sometimes even up to 90%, indicating that the consistency of Emb-size across searching is considerably higher than AutoDim and other methods. In addition, we found that AutoEmb is more inclined to choose a larger Emb-size for all features, while ESAPN tends to choose smaller ones. In contrast, AdaS&S does not have such issue, it assigns the most suitable Emb-size for each feature, e.g., the “Gender” feature is always set to smallest Emb-size and the “MovieID” always set to the largest. The analysis above demonstrates that AdaS&S has better robustness compared with past AES methods, the Criteo and Avazu dataset also yields a similar conclusion.

Table 3: Overall Performance of AdaS&S. AdaS&S-R is the resource-first mode. AdaS&S-E is the effect-first mode. P-R indicates the percentage of Parameters Reduction compared to UES-32. Higher AUC or lower LogLoss mean better performance. *** indicates p-value < 0.001 in significance test (two-sided t-test) with the baseline in same order.

Dataset	Base Model	Metric	Baseline		Past AES Method				Our Framework		
			UES-32	UES-24	MDE	AutoDim	AutoEmb	ESAPN	SSEDS	AdaS&S-R	AdaS&S-E
Avazu	Wide&Deep	AUC	0.7811	0.7784	0.7806	0.7821	0.7818	0.7810	0.7824	0.7822	0.7846*
		LogLoss	0.3872	0.3915	0.3894	0.3859	0.3864	0.3878	0.3861	0.3860	0.3842*
		P-R	--	27.31%	23.45%	8.43%	14.13%	25.51%	24.09%	32.97%	21.58%
	DeepFM	AUC	0.7829	0.7803	0.7813	0.7837	0.7841	0.7831	0.7837	0.7842	0.7866*
		LogLoss	0.3849	0.3880	0.3865	0.3839	0.3831	0.3845	0.3834	0.3828	0.3807*
		P-R	--	26.19%	24.93%	12.17%	10.41%	23.94%	23.27%	31.12%	20.83%
ML-1M	Wide&Deep	AUC	0.7873	0.7846	0.7868	0.7881	0.7879	0.7870	0.7880	0.7882	0.7905*
		LogLoss	0.5568	0.5604	0.5585	0.5547	0.5558	0.5567	0.5554	0.5550	0.5532*
		P-R	--	24.91%	21.93%	6.10%	17.27%	24.24%	23.57%	31.71%	22.71%
	DeepFM	AUC	0.7891	0.7867	0.7886	0.7898	0.7904	0.7893	0.7899	0.7901	0.7923*
		LogLoss	0.5547	0.5572	0.5561	0.5539	0.5530	0.5542	0.5532	0.5532	0.5513*
		P-R	--	24.54%	23.29%	11.57%	8.81%	22.72%	22.83%	30.68%	21.05%
Criteo	Wide&Deep	AUC	0.7984	0.7955	0.7977	0.7990	0.7988	0.7981	0.7992	0.7995	0.8017*
		LogLoss	0.4935	0.4964	0.4944	0.4923	0.4923	0.4932	0.4920	0.4919	0.4901*
		P-R	--	29.07%	28.51%	17.94%	16.03%	26.59%	27.54%	34.10%	25.12%
	DeepFM	AUC	0.8013	0.7987	0.8005	0.8031	0.8027	0.8011	0.8034	0.8030	0.8044*
		LogLoss	0.4892	0.4918	0.4899	0.4873	0.4878	0.4893	0.4870	0.4876	0.4860*
		P-R	--	28.53%	27.14%	14.03%	11.40%	23.48%	25.74%	32.81%	23.94%

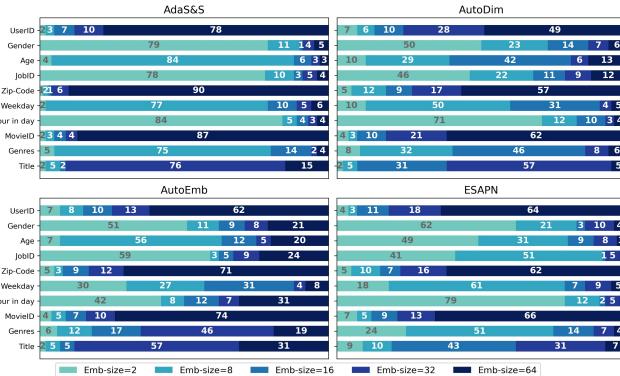


Figure 4: Stability of AES. Each row represents the number of times a feature is assigned to various Emb-sizes in 100 searches.

4.4.2 Ablation Study: Resources Competition Penalty (RQ3).

In order to demonstrate the effect of resource competition penalty in AdaS&S, we conduct more experiments and analyze their results. With DeepFM as the DLRM, on ML-1M dataset we search the Emb-size assignment with different weights (λ_r and λ_c in Eq.(9)) in the resources competition penalty, then we retrain the model using the produced Emb-sizes. Performances of retrained models is shown in Fig.5. To obtain convincing results, all experiments are conducted for multiple times with various random seeds, the mean of metrics and standard error of the mean is reported.

According to the left side of Fig.5, the larger λ_r is, the more parameters is reduced. At the same time, the AUC metric drops and leads to worse recommendation effect. This indicates that larger λ_r will result in stronger limitation on memory resources. The right side of Fig.5 shows that, when $\lambda_c < 0.05$, AUC and P-R grows together with λ_c . When $\lambda_c > 0.05$, AUC will decline as λ_c keeps increasing, and the growth of P-R also stopped after

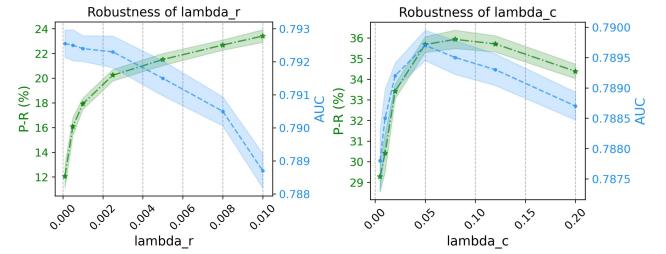


Figure 5: Effect of resources competition penalty. P-R indicates the percentage of Parameters Reduce. Left part is the robustness test of λ_r , where λ_c is set as 0.08. Right part is the robustness test of λ_c where λ_r is set as 0.005.

$\lambda_c > 0.08$. This indicates that larger λ_c will result in fiercer competition between features. Features that are in a disadvantaged position in competition will be assigned a smaller Emb-size, yet it may need a larger Emb-size to fully represent the information within. Therefore, the recommendation effect will degrade if λ_c is too large. In summary, the resource competition penalty (Eq.(9)) can balance the computation resources and recommendation effectiveness during AES, guide the AES results to suit specific resource budget. In practice, λ_r shall be adjusted manually based on actual requirements, and a moderate λ_c is recommended.

4.4.3 Ablation Study: Supernet Training (RQ4).

The consistency of the trained supernet is crucial for obtaining a proper and stable search results (Section.2.3). We utilize Kendall-tau correlation of architecture performance ranking based on subnet (inherited weights) and their ground truths (stand-alone trained weights) [26, 34]. Note that Kendall-tau value (dubbed as “KD”) is in $[-1, 1]$ and higher value indicates better supernet consistency. In particular, we randomly select 200 different subnet architectures (Emb-size allocations) to evaluate the KD value of AUC and LogLoss. The empirical results are shown in Table 4.

Table 4: Consistency of supernet.

Scheme	Sampling	KD(LogLoss)	KD(AUC)
Shared	Random	0.7246	0.7020
Shared	Vanilla&Uniform	0.7488	0.7351
Shared	Weight&Uniform	0.7541	0.7409
Shared	Adaptive	0.7891	0.7815
Indep	Random	0.7384	0.7412
Indep	Vanilla&Uniform	0.7498	0.7455
Indep	Weight&Uniform	0.7673	0.7547
Indep	Adaptive	0.7904	0.7970

In Table 4, “Indep” is for the independent supernet scheme and “Shared” for the shared scheme. We also conduct ablation study on different training methods and compare the consistency: in the “Sampling” column, “Random” is the naive sampling method for training subnets similar to [12]. At each training step, it selects a feature with the probability $p = 0.6$, and then picks a candidate embedding of the feature with $p = 1/T$. The “Vanilla&Uniform” method is from OptEmbed [33], for each feature, it assumes the probability for training each dimension as a uniform distribution, i.e., sampling rate for candidate embeddings are proportional to their Emb-sizes. “Weight&Uniform” further selects features by weighted probabilities [21], features with larger cardinality will have larger weight to be sampled, and the sampling for candidate Emb-sizes is same as “Vanilla&Uniform”. The “Adaptive” method is our proposed solution in Section 3.1.

From Table 4, we can conclude that the “Indep” scheme performs better than “Shared”, and we attribute this to the fact that there exists overlapping between different embedding tables in the “Shared” scheme. Note that given same sampling method, the “Shared” scheme is not much worse than “Indep”. Further, the adaptive supernet training outperforms all other training methods. In summary, our adaptive supernet is of satisfactory consistency and could better guide the searching stage.

4.4.4 Efficiency Analysis (RQ5).

Apart from model effectiveness, efficiency is also key consideration in real-world applications. In this section, we investigate the influence of AdaS&S on efficiency of AES and model inference. Specifically, we compare the training time of AES methods (without the model retraining) to show their searching efficiency. And we summarize Floating-point Operations (FLOPs) of the corresponding retrained model in searched Emb-sizes to indicate the inference efficiency. The experiment is conducted on Criteo dataset using Wide&Deep architecture with one Tesla T4 GPU, we visualize the results in Figure 6.

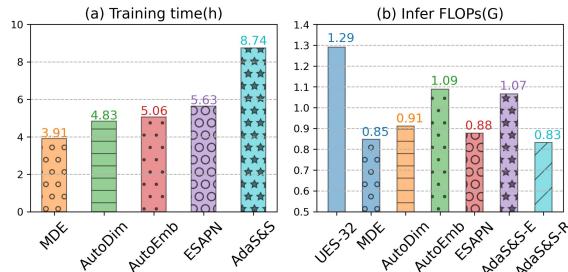


Figure 6: Training time and FLOPs of AES methods. Left side shows training time of each AES methods. Right side compares FLOPs with unified Emb-size models (UES-32).

In Fig.6(a), AdaS&S requires highest training time to produce AES result, and we attribute this to the two-stage design of our framework, which inevitably add additional time costs (note that we show the total time cost of two stages together for AdaS&S). It is worth mentioning that though total cost is higher, our approach benefits from the one-shot nature and the search stage is more light-weighted than supernet training stage, so the cost of AdaS&S is much lower than others when conducting multiple times of searching for various requirements once the supernet is trained.

From Fig.6(b), AES methods could improve the inference efficiency because of the parameter reduction (illustrated in Table 3). Despite using the effect-first mode, AdaS&S-E could still achieve FLOPs comparable to other methods. The resource-first mode (AdaS&S-R) significantly reduce the inference overhead with lowest FLOPs. In summary, the superior efficiency of AdaS&S makes it quite suitable to be deployed in real-world tasks.

5 RELATED WORK

AutoML Techniques: As a core component of AutoML, Neural Architecture Search(NAS) has raised research interests since [29]. To address the computation overhead, some researches [35–37] utilize parameter sharing to search sub-graphs from a large network graph. Bender et al. [38] systematically formulate the supernet method and identify the effectiveness of weight sharing. Guo et al. [12] improve the supernet optimization with uniform path sampling. FairNAS [21] further propose the strict fairness sampling method. Other studies [11, 11, 39, 40] model the network architecture into continuous search space. With differentiable searching, these methods can optimize with gradient descent.

AES methods: Heuristic, NAS-based, and pruning methods are primary categories of AES methods. As a heuristic method, MDE [4] set Emb-sizes based on popularity of features. However, these rule-based solutions are not able to incorporate with task-specific models, and the performance is not guaranteed [3]. Recent studies [2, 8, 9, 41, 42] introduce NAS into AES. AutoEmb [8] selects Emb-sizes with a controller network and update it together with main network by bi-level optimization in DARTS [11]. AutoDim [2] utilizes the DARTS framework to optimize weights over different Emb-sizes. NIS [41] searches the optimal embeddings for each feature value in a discrete space, and parameters for embeddings and controller are optimized jointly. These methods can hardly guide searching results and meet specific resource demands, and the jointly optimizing of embedding training and searching still needs further study. Other methods would prune embedding parameters to smaller Emb-sizes. For example, AMTL [3] introduces a twins-based layer to generate mask vectors to prune embeddings. PEP [5] removes redundant dimensions with the help of a learnable threshold for embedding values. SSEDS [6] prunes Emb-sizes by measuring the influence of each dimensions on the loss. The one-shot supernet for embeddings in OptEmbed [33] is similar with ours, yet it trains supernet with uniform sampling and searches dimension masks by Evolution Algorithm. The learning of pruning thresholds or masks in AES methods above raises the risk of unstable searching results, which also undermines the interpretability of outcomes.

6 CONCLUSION

In this paper, we propose a novel automatic embedded size search (AES) method called AdaS&S. Specifically, we design a one-shot supernet framework to tackle the AES problem. First, we decouple main model training from the search process, which ensures the soundness of parameters and stability of Emb-size allocation. We design the Adaptive Sampling method for training supernet, which greatly improves its consistency. Second, in the searching stage, we devise a RL-based method to seek the optimal AES result. Finally, a well-designed resource competition penalty is employed to balance the performance and the resources cost. We evaluate the AdaS&S framework with extensive experiments on widely used datasets, showing that our framework can outperform other AES methods and address the key challenges in past research. The source code of the proposed framework is available online⁶.

In the future, we would like to further study the training efficiency of AdaS&S and try to reduce the total time cost needed for two stages. We will also explore more ways to leverage the prior knowledge about the recommendation tasks and features to improve the model effectiveness.

REFERENCES

- [1] 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. *ACM Transactions on Information Systems (TOIS)*, 37(1):1–35, 2018.
- [2] Xiangyu Zhao, Haochen Liu, Hui Liu, Jiliang Tang, Weiwei Guo, Jun Shi, Sida Wang, Huiji Gao, and Bo Long. Autodim: Field-aware embedding dimension search in recommender systems. In *Proceedings of the Web Conference 2021*, pages 3015–3022, 2021.
- [3] Bencheng Yan, Pengjie Wang, Kai Zhang, Wei Lin, Kuang-Chih Lee, Jian Xu, and Bo Zheng. Learning effective and efficient embedding via an adaptively-masked twins-based layer. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 3568–3572, 2021.
- [4] Antonio A Ginart, Maxim Naumov, Dheevatsa Mudigere, Jiyan Yang, and James Zou. Mixed dimension embeddings with application to memory-efficient recommendation systems. In *2021 IEEE International Symposium on Information Theory (ISIT)*, pages 2786–2791. IEEE, 2021.
- [5] Siyi Liu, Chen Gao, Yihong Chen, Depeng Jin, and Yong Li. Learnable embedding sizes for recommender systems. *arXiv preprint arXiv:2101.07577*, 2021.
- [6] Liang Qu, Yonghong Ye, Ningzhi Tang, Lixin Zhang, Yuhui Shi, and Hongzhi Yin. Single-shot embedding dimension search in recommender system. *arXiv preprint arXiv:2204.03281*, 2022.
- [7] Shuming Kong, Weiyu Cheng, Yanyan Shen, and Linpeng Huang. Autosrh: An embedding dimensionality search framework for tabular data prediction. *IEEE Transactions on Knowledge and Data Engineering*, 2022.
- [8] Xiangyu Zhao, Haochen Liu, Wenqi Fan, Hui Liu, Jiliang Tang, Chong Wang, Ming Chen, Xudong Zheng, Xiaobing Liu, and Xiwang Yang. Autoemb: Automated embedding dimensionality search in streaming recommendations. In *2021 IEEE International Conference on Data Mining (ICDM)*, pages 896–905. IEEE, 2021.
- [9] Haochen Liu, Xiangyu Zhao, Chong Wang, Xiaobing Liu, and Jiliang Tang. Automated embedding size search in deep recommender systems. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2307–2316, 2020.
- [10] Pengyu Zhao, Kecheng Xiao, Yuanxing Zhang, Kaigui Bian, and Wei Yan. Ameir: Automatic behavior modeling, interaction exploration and mlp investigation in the recommender system. In *IJCAI*, pages 2104–2110, 2021.
- [11] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.
- [12] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In *European conference on computer vision*, pages 544–560. Springer, 2020.
- [13] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)*, 52(1):1–38, 2019.
- [14] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Rishabh Iyer, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pages 7–10, 2016.
- [15] Huirong Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: a factorization-machine based neural network for ctr prediction. *arXiv preprint arXiv:1703.04247*, 2017.
- [16] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, pages 191–198, 2016.
- [17] 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 *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1059–1068, 2018.
- [18] Rita Yi Man Li, Beiqi Tang, and Kwong Wing Chau. Sustainable construction safety knowledge sharing: A partial least square-structural equation modeling and a feedforward neural network approach. *Sustainability*, 11(20):5831, 2019.
- [19] Steffen Rendle. Factorization machines. In *2010 IEEE International Conference on Data Mining*, pages 995–1000, 2010.
- [20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- [21] Xiangxiang Chu, Bo Zhang, and Ruijun Xu. Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12239–12248, 2021.
- [22] Taehyeon Kim and Se-Young Yun. Supernet training for federated image classification under system heterogeneity. *arXiv preprint arXiv:2206.01366*, 2022.
- [23] Minghao Chen, Jianlong Fu, and Haibin Ling. One-shot neural ensemble architecture search by diversity-guided search space shrinking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16530–16539, 2021.
- [24] Shan You, Tao Huang, Mingmin Yang, Fei Wang, Chen Qian, and Changshui Zhang. GreedyNAS: Towards fast one-shot nas with greedy supernet. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1999–2008, 2020.
- [25] Han Cai, Chuang Gan, Tianzhe Wang, Zhehai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*, 2019.
- [26] Kaicheng Yu, Christian Sciuто, Martin Jaggi, Claudiu Musat, and Mathieu Salzmann. Evaluating the search phase of neural architecture search. *arXiv preprint arXiv:1902.08142*, 2019.
- [27] Ghada Sokar, Zahra Atashgahi, Mykola Pechenizkiy, and Decebal Constantin Mocanu. Where to pay attention in sparse training for feature selection? *arXiv preprint arXiv:2211.14627*, 2022.
- [28] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.
- [29] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.
- [30] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.
- [31] He Wei, Yuekui Yang, Haiyang Wu, Yangyang Tang, Meixi Liu, and Jianfeng Li. Automatic feature selection by one-shot neural architecture search in recommendation systems. In *Proceedings of the ACM Web Conference 2023*, pages 1765–1772, 2023.
- [32] Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. Autoint: Automatic feature interaction learning via self-attentive neural networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1161–1170, 2019.
- [33] Fuyuan Lyu, Xing Tang, Hong Zhu, Huifeng Guo, Yingxue Zhang, Ruiming Tang, and Xue Liu. Optembed: Learning optimal embedding table for click-through rate prediction. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 1399–1409, 2022.
- [34] Miao Zhang, Huiqi Li, Shirui Pan, Taoping Liu, and Steven W Su. One-shot neural architecture search via novelty driven sampling. In *IJCAI*, pages 3188–3194, 2020.
- [35] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In *International conference on machine learning*, pages 4095–4104. PMLR, 2018.
- [36] Ruiqi Zheng, Liang Qu, Bin Cui, Yuhui Shi, and Hongzhi Yin. Automl for deep recommender systems: A survey. *ACM Transactions on Information Systems*, 2023.
- [37] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)*, 52(1):1–38, 2019.
- [38] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and simplifying one-shot architecture search. In *International conference on machine learning*, pages 550–559. PMLR, 2018.
- [39] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *Proceedings of the 2023 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '23)*, pages 103–112, 2023.

⁶<https://github.com/XXXXXX/XXXXXX>

- the IEEE International Conference on Computer Vision*, pages 1294–1303, 2019.
- [40] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. Snas: stochastic neural architecture search. *arXiv preprint arXiv:1812.09926*, 2018.
 - [41] Manas R Joglekar, Cong Li, Mei Chen, Taibai Xu, Xiaoming Wang, Jay K Adams, Pranav Khaitan, Jiahui Liu, and Quoc V Le. Neural input search for large scale recommendation models. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2387–2397, 2020.
 - [42] Weiyu Cheng, Yanyan Shen, and Lipeng Huang. Differentiable neural input search for recommender systems. *arXiv preprint arXiv:2006.04466*, 2020.