AutoField: Automating Feature Selection in Deep Recommender Systems

Yeijing Wang¹, Xiangyu Zhao¹∗, Tong Xu², Xian Wu³
¹City University of Hong Kong, ²University of Science and Technology of China, ³Tencent Inc
adave631@gmail.com,xianzhao@cityu.edu.hk,tongxu@ustc.edu.cn,kevinxwu@tencent.com

Abstract
Feature quality has an impactful effect on recommendation performance. Thereby, feature selection is a critical process in developing deep learning-based recommender systems. Most existing deep recommender systems, however, focus on designing sophisticated neural networks, while neglecting the feature selection process. Typically, they just feed all possible features into their proposed deep architectures, or select important features manually by human experts. The former leads to non-trivial embedding parameters and extra inference time, while the latter requires plenty of expert knowledge and human labor effort. In this work, we propose an AutoML framework that can adaptively select the essential feature fields in an automatic manner. Specifically, we first design a differentiable controller network, which is capable of automatically adjusting the probability of selecting a particular feature field; then, only selected feature fields are utilized to retrain the deep recommendation model. Extensive experiments on three benchmark datasets demonstrate the effectiveness of our framework. We conduct further experiments to investigate its properties, including the transferability, key components, and parameter sensitivity.

CCS Concepts
• Information systems → Recommender systems.

Keywords
Feature Selection, Recommender System, AutoML

ACM Reference Format:
Yeijing Wang¹, Xiangyu Zhao¹∗, Tong Xu², Xian Wu³. 2022. AutoField: Automating Feature Selection in Deep Recommender Systems. In Proceedings of the ACM Web Conference 2022 (WWW ’22), April 25–29, 2022, Virtual Event, Lyon, France. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3485447.3512071

1 INTRODUCTION
In the world-wide-web era, customers are overwhelmed with overloading information, making it difficult to retrieve valuable information. Well-designed recommender systems, which significantly alleviate this issue by presenting selected items that best match user preference [35, 36], are widely employed in a variety of scenarios, including shopping websites [34, 39], online movie/music platforms [49, 50], and location-based services [28].

With the rapid growth of deep learning theories and techniques, deep learning-based recommender systems could capture user preference accurately with their strong feature representation and inference capacities [44]. The majority of existing works spend lots of effort on designing novel and sophisticated neural architectures; however, feature selection, the process of selecting a subset of informative and predictive feature fields, has received little attention [2, 8, 30]. They typically feed their proposed deep recommender systems with all possible features without dropping the unavailing ones [11, 14, 15]. This often necessitates additional computations to learn all the feature embeddings, thereby further slowing down the model inference. Sometimes, ineffective features can even be detrimental to model performance [31].

Several conventional feature selection methods can be leveraged in developing deep recommender systems. First, hand-crafted feature selection by human experts is generally used in traditional recommender systems. In fact, this is a trial-and-error method manually conducted by human experts, which requires necessary expert knowledge and laborious human efforts. In addition, grid search [9, 16] or random search [3] algorithms, which exhaustively or randomly sample the solutions from all possible feature selection options, are comparatively accurate but prohibitively expensive in computations and time. Statistics and traditional machine learning techniques can also be utilized in feature selections, such as filter methods via correlation analysis [13, 43], as well as embedded
methods via Lasso regularization [7] and decision trees [19]. However, these methods usually fail in deep recommender systems. For instance, filter methods neglect the dependency between feature selection and downstream deep recommendation models; Embedded methods are sensitive to the recommendation models’ strong structural assumptions. With recent advances in automated machine learning (AutoML) [42], there is a tremendous interest in developing deep recommender systems via AutoML technologies. However, most efforts aim to select predictive feature interactions [23, 29, 37], while few of them focus on selecting predictive features directly.

In this paper, we aim to propose an efficient framework that automates the feature selection process when building deep recommender systems. This task is rather challenging due to the huge search space of all feature selection options, i.e., $2^N$ with $N$ the number of feature fields. To address this issue, we develop an AutoML-based framework AutoField, which can automatically identify an optimal subset of predictive feature fields from the whole search space. In addition, AutoField is capable of processing discrete categorical features and simultaneously evaluating the contributions of all feature fields in a parallel fashion. The experiment results on three benchmark datasets demonstrate the effectiveness of our proposed model. We summarized our major contributions as follows:

- We propose a novel AutoML-based framework, AutoField, which can select predictive features automatically, improving the model performance and inference efficiency;
- We design a controller architecture to dynamically moderate the probability pairs deciding whether to select or drop feature fields. Decisions are instructive for other deep recommender systems on the same dataset;
- We carry out extensive experiments on real-world benchmark datasets with investigating the effectiveness, efficiency, and transferability of the proposed framework.

We organize the rest of this paper as follows. In Section 2, we present the main modules of the proposed framework, the optimization strategy, and the retraining method. Section 3 details the experiments, including the research questions we care about and corresponding experimental results. In Section 4, we introduce related works in brief. Finally, we conclude this work in Section 5.

2 THE PROPOSED FRAMEWORK

In this section, we introduce the technical details of the proposed AutoField framework. We first provide an overview of AutoField. And then, we detail the important modules, the AutoML-based optimization algorithm, and the retraining method.

2.1 Framework Overview

In this subsection, we will provide an overview of the AutoField framework. AutoField aims to select the optimal subset of feature fields from the entire feature space automatically. The framework overview is illustrated in Figure 2. Briefly, the proposed framework consists of two stages, the search stage, and the retraining stage. To find the optimal subset of feature fields, we first update the framework parameters in the search stage. Then, we select the best feature fields based on learned parameters and feed them into subsequent deep recommender systems in the retraining stage.

In the search stage, we first initialize the framework parameters. Then, we feed all the feature fields to the feature selection module, which bridges the controller with the deep recommendation model.
by selecting feature fields in a differentiable manner according to the controller parameters. After that, the deep recommendation model predicts the user preference. With prediction errors on the training set, we update the parameters of the deep recommendation model. Simultaneously, the controller is optimized based on validation data. At the end of this stage, we obtain the well-trained parameters in the controller for the next stage.

All possible feature fields are fed into the deep recommendation model during the search stage, but only a part of the feature fields will be selected for the final deep recommendation model. As a result, we retrain the model after the search stage. Specifically, the feature selection module first selects the predictive feature fields based on the controller parameters. To be noted, the behavior of feature selection module is different in two stages. We conduct hard selection in the retraining stage but use soft selection in the search stage. Subsequently, with selected feature fields, the deep recommendation model is adapted and retrained.

2.2 Deep Recommendation Model

We will introduce the architecture of the deep recommendation model in this subsection. As visualized in Figure 3, a typical deep recommendation model has two essential components: the embedding layer and the MLP layer.

2.2.1 Embedding Layer. The embedding layer is a basic component widely used in deep recommender systems, which converts the categorical inputs into low-dimensional continuous vectors via two steps, i.e., binarization and projection.

Binarization. Categorical inputs are usually transformed into binary vectors. The dimension of these vectors is equivalent to the number of unique feature values in the corresponding feature fields. For example, the feature field “Gender” has two unique feature values, “Male” and “Female”, then they can be represented by the two-dimensional binary vectors. The dimension of these vectors is equivalent to the number of unique feature values in the corresponding feature fields.

Projection. Then, given N input feature fields of a user-item interaction data, the N features can be represented as the concatenation of their binary vectors, i.e., \( x = [x_1, x_2, \cdots, x_N] \), \( x_n \in \mathbb{R}^{D_n} \), where \( x_n \) is the binary vector for the \( n^{th} \) feature field, and \( D_n \) is the corresponding dimension.

Due to the high sparsity and varied length, binary vectors still require further processing in the projection step. For \( x_n, \forall n \in [1, N] \), we project it into low-dimensional space as:

\[
e_n = A_n x_n
\]

where \( A_n \in \mathbb{R}^{d \times D_n} \) is a learnable weight matrix, and \( d \) is the predefined embedding size of the projection space. Then, the final embedding of the user-item interaction data is:

\[
E = [e_1, e_2, \cdots, e_N]
\]

2.2.2 MLP Layer. Multi-layer Perceptron (MLP) structure is a common component in deep recommender systems [11, 34]. It mines information from feature embedding by linear transformations and non-linear activations, formulated as:

\[
h_0 = E
\]

\[
h_{m+1} = \text{ReLU}(W_m h_m + b_m), 0 \leq m \leq M
\]

\[
\hat{y} = \sigma(W_M h_M + b_M + 1)
\]

where the initial input of MLP, say \( h_0 \), is the output of the embedding layer. For Equation (4), \( W_m, b_m, h_m \) respectively stand for the weight matrix, bias vector, and output of \( m^{th} \) layer, where \( M \) is the number of MLP layers before the output layer. For Equation (5), \( W_{M+1}, b_{M+1} \) are the weight matrix and bias vector of the output layer, and \( \sigma(\cdot) \) is the activation function depending on specific recommendation tasks.

2.3 Controller

In this subsection, we will introduce our feature selection’s search space and the controller structure. For an input dataset with \( N \) feature fields, we have two choices for each feature field, i.e., “selected” or “dropped”, thus the size of the whole search space is \( 2^N \). With this huge space, common search space construction methods like encoding [33] are incompatible with the practical demand for efficiency. Motivated by [24], we define our search space as a directed acyclic graph in Figure 4. To be more specific, we utilize \( N \) parallel nodes to select \( N \) feature fields respectively, where the \( n^{th} \) node is a 2-dimensional vector containing two parameters (\( a_n^1, a_n^0 \)). In other words, there are \( 2 \times N \) parameters in total for the \( N \) feature fields, which control the behaviors of the feature selection process. We denote \( a_n^1 \) as the probability of selecting a feature field and \( a_n^0 \) as that of dropping the feature field, thus we have \( a_n^1 + a_n^0 = 1 \).

As visualized in Figure 4 (a), we first assign a pair of equivalent (\( a_n^1, a_n^0 \)) to each feature field, i.e., \( a_n^1 = a_n^0 = 0.5 \). During the training progress as in Figure 4 (b), the parameter \( a_n^1 \) of predictive features (e.g., Field 1 and 3) will increase, while \( a_n^0 \) will decrease. For non-predictive features (e.g., Field 2), the situation is the opposite. Finally, as in Figure 4 (c), the feature fields with higher probability \( a_n^1 \) will be selected for model retraining, while the ones with higher \( a_n^0 \) will be dropped.
Therefore, a hard selection method is highly desirable. To address the hard selection process based on the controller’s parameters, we introduce the Gumbel-Max trick \cite{10} to simulate the influence of suboptimal feature fields on the final recommendation task. We denote the embedding of the feature field in the search stage as $\mathbf{e}_n$, each feature field will be selected or dropped for the final recommendation of AutoField in the search stage. Since $\log \alpha_n^u + g_0$, $\log \alpha_n^u + g_1$ in Equation (8) is a pair of extreme values, i.e., $p_n^u$ is very close to 1 or 0, then we can use $p_n^j$ to simulate the hard selection, which bridges the aforementioned gap between model search and retraining stages.

Finally, the feature selection in Equation (6) can be rewritten as below:

$$e_n^j = (p_n^j \cdot 1 + p_n^0 \cdot 0) \cdot e_n = p_n^j e_n$$ \hspace{1cm} (10)

Based on Equation (10), we can obtain the $E'$ as in Equation (7), and then replace the embeddings $E$ of Equation (2) by $E'$, so as to conduct feature selection with the proposed controller.

### 2.5 Optimization

In the above subsections, we introduce the architecture of the deep recommendation model, the controller, and formulate the feature selection process based on controller parameters. Next, we will detail the optimization.

In the search stage, there are two sets of parameters to be optimized, i.e., those of the deep recommendation model and the controller. We denote the parameters of the deep recommendation model as $W$ and the parameters of the controller as $A$. $W$ and $A$ should not be jointly learned from the same batch of training data as traditional supervised learning \cite{33}. This is because the learning process of $W$ and $A$ are highly interdependent, which would lead to a severe over-fitting problem if we update both of them on the same training batch.

Inspired by DARTS \cite{24}, we formulate a bi-level optimization problem \cite{1} to tackle the over-fitting issue by updating $A$ and $W$ alternatively based on different sets of data:

$$\min_A \mathcal{L}_{val} (W^*(A), A)$$

s.t. $W^*(A) = \arg \min_W \mathcal{L}_{train} (W, A^*)$ \hspace{1cm} (11)

where $\mathcal{L}_{train}$ and $\mathcal{L}_{val}$ are the binary cross-entropy loss (BCE)\(^1\) $(y, \hat{y}) = y \cdot \log \hat{y} + (1 - y) \cdot \log (1 - \hat{y})$ on the training set and $\mathcal{L}_{val}$ on the validation set. To be specific, we can draw a hard selection $z_n$ as:

$$z_n = \text{one}_\text{hot} \left( \arg \max \{ \log \alpha_n^u + g_0, \log \alpha_n^u + g_1 \} \right)$$ \hspace{1cm} (8)

where $g_j = -\log (u_j)$

$u_j \sim \text{Uniform}(0, 1)$ $\forall j \in [0, 1]$ where $(g_j)$ are independent and identically distributed (i.i.d) gumbel noises. For $n^{th}$ feature field, if we define the selection behavior as $z_n \cdot \mathbf{e}_n$, then its expectation is equivalent to the weighted sum version as in Equation (6).

Note that $z_n$ in Equation (8) is obtained from the non-differentiable operation, i.e., $\text{one}_\text{hot} (\arg \max (\cdot))$. To apply the gradient optimization strategy, we approximate it by the softmax operation \cite{17} to make the architecture differentiable as:

$$p_n^j = \frac{\exp \left( (\log \alpha_n^j + g_j) / \tau \right)}{\exp \left( (\log \alpha_n^j + g_j) / \tau \right) + \exp \left( (\log \alpha_n^0 + g_0) / \tau \right)}$$ \hspace{1cm} (9)

where $p_n^j$ is the new probability for selecting ($j = 1$) or dropping ($j = 0$) the $n^{th}$ feature, and $\tau$ is a hyperparameter called temperature.
validation set, respectively, \( y \) are the ground truth, and \( \hat{y} \) are the model predictions.

The detailed optimization of the AutoField framework is illustrated in Algorithm 1. We manually set hyper-parameters: (i) the number of finally selected feature field \( K \); and (ii) the frequency \( f \) of updating controller parameters \( A \), compared with that of \( W \). In other words, we train \( A \) once after training \( W \) of \( f \) times. To be specific, we first set a flag \( t \) to record the training steps (line 1). Then, we train \( W \) and \( A \) to converge (line 2), where we train the recommendation model parameters \( W \) with every mini-batch of training data (line 3-4) and train the controller parameters with a mini-batch of validation data every \( f \) time steps (line 5-8).

### Algorithm 1 Optimization Algorithm of AutoField Framework

**Input:** Raw feature fields \( \{f_n\} \), number of selected feature fields \( K \), controller update frequency \( f \)  
**Output:** \( K \) selected feature fields

1. \( t = 0 \) (\( t \) records the training steps)
2. while not converge do
3. Sample a mini-batch from the training dataset
4. Update \( W \) according to the constraint in Equation (11)
5. if \( t \% f = 0 \) then
6. Sample a mini-batch from the validation dataset
7. Update \( A \) according to the Equation (11)
8. end if
9. \( t = t + 1 \)
10. end while

---

**3 EXPERIMENT**

In this section, we conduct extensive experiments to evaluate the effectiveness of the proposed framework. Specifically, the main research questions we care about are as follows:

- **RQ1:** How does AutoField perform compared with other feature selection methods?
- **RQ2:** Are the selected feature fields of AutoField transferable?
- **RQ3:** What is the impact of the components in AutoField?
- **RQ4:** What is the influence of key hyper-parameters?
- **RQ5:** Does AutoField really select out the optimal subset of feature fields?

#### 3.1 Datasets

We evaluate the overall performance of the AutoField framework on two benchmark datasets:

- **Avazu:** This dataset was provided for a Kaggle click-through rates (CTR) prediction competition. There are 40 million user-click records in this dataset with 22 feature fields. Parts of them are anonymous.
- **Criteo:** This is a real-world industry benchmark dataset for predicting the CTR of online advertisements. It includes 45 million user-click records on delivered ads with 26 categorical feature fields and 13 numerical feature fields. We convert the numerical feature fields to categorical\(^3\). All feature fields are anonymous.

For both datasets, we use 80% for training, 10% for validation, and the rest 10% for testing.

#### 3.2 Evaluation metric

We use the AUC score (Area Under the ROC Curve) and Logloss to evaluate recommendation performance. It is noteworthy that a higher AUC score and lower Logloss at 0.001-level could indicate a significant improvement in click-through rates prediction tasks [11]. Besides, to compare the efficiency of models, we record the inference time per batch and the total retraining time of models with selected feature fields.

#### 3.3 Implementation

In this subsection, we detail the implementation of AutoField framework. For the embedding layer, we set the embedding size of all feature fields as 16. For the MLP layer, we use two fully-connected layers of size \([16, 8]\) and use the activation function ReLU. Since the click-through rates prediction task could be regarded as a binary classification problem, we use the activation function Sigmoid in the output layer. For the controller, as introduced in Section 2.3, we assign two parameters \((a^1_n, a^2_n)\) to \(n\)th feature field and apply Gumbel–Softmax to simulate hard selection. For other parameters, we set all learning rates as 0.0001, the batch size as 2048, and the drop-out rate as 0.2. The temperature in Equation (8) is \(\tau = \max(0.01, 1 - 0.00005 \cdot t)\), where \(t\) is the training step.

Note that our implementation is based on a public Pytorch library for recommendation models\(^4\). Feature selection baselines are

---

\(^2\)https://www.kaggle.com/c/avazu-ctr-prediction/  
\(^3\)https://www.kaggle.com/c/criteo-display-ad-challenge/  
\(^4\)https://www.csie.ntu.edu.tw/~r01922136/kaggle-2014-criteo.pdf  
\(^5\)https://github.com/rixwew/pytorch-fm
3.4 Overall Performance (RQ1)

To evaluate the effectiveness of AutoField, we conduct extensive experiments with different feature selection methods based on the deep recommendation model mentioned in Section 2.2. The representative baselines include PCA [40], LASSO [38], GBR [4], GBDT [4] and IDARTS [18]. Details of baseline settings can be found in Appendix A. AutoField drops 11 feature fields for Avazu (50%) and 6 feature fields for Criteo (15%). For a fair comparison, baselines also drop the same number of feature fields (or less). The overall performance is shown in Table 1. We can find that:

- For both datasets, applying AutoField to select predictive features could enhance the recommendation performance. It is noteworthy that we dropped 50% of the feature fields in Avazu, but achieve better performance.

- PCA and LASSO can select some informative feature fields like AutoField. However, they miss some other predictive feature fields and select some trivial feature fields. The main reason is that PCA neglects the correlations between ground truth $y$ and feature fields, while LASSO is sensitive to the strong structural assumptions of the deep recommendation model.

- For gradient boosting methods, i.e., GBR and GBDT, they obtain a better selection result on Criteo than the two previous baselines. However, GBDT gets an improper selection on Avazu. The performance of gradient boosting methods, which assign importance weights to feature fields, is highly dependent on the predicting ability of origin classifiers. The original GBDT method (without MLP layer) only achieves an AUC of 0.52 on Avazu, which indicates its ineffectiveness as an original classifier. Besides, the effectiveness of gradient boosting methods lies in the ensemble structures, which require lots of computations.

- Another AutoML-based method IDARTS cannot generate stable selection results, so we show its average performance of various selection results in Table 1. The reason is that IDARTS considers all feature fields in a single search space and applies softmax over all of them, which might be significantly disturbed by the stochastic training process and stuck at the local optimum. AutoField introduces Gumbel-Softmax to prevent local optimum.

- Besides, with selected features, the retraining time of the deep recommendation model can be reduced significantly, such as 50% on the Avazu dataset.

| Selection Method | Dropped Feature Fields | AUC↑ | Logloss↓ |
|------------------|------------------------|------|---------|
| All Fields       | None                   | 0.8029 | 0.4490 |
| PCA              | [8, 10, 16, 20, 28, 33] | 0.7999 | 0.4516 |
| LASSO            | [9, 17, 18, 20, 21]    | 0.8000 | 0.4514 |
| GBR              | [13, 17, 20]           | 0.8022 | 0.4497 |
| GBDT             | [7, 12, 16, 19, 21, 23]| 0.8010 | 0.4506 |
| IDARTS           | Unstable               | 0.7985 | 0.4518 |
| AutoField        | [13, 14, 17, 20, 21, 33] | 0.8029 | 0.4490 |
|                  | [1, 5, 7, 8, 12, 13, 15, 16, 18, 19, 21] | 0.7773 | 0.3813 |

Table 2: Transferability analysis on Avazu

| Model      | AUC   | Logloss | Inference/ms | Time Saved |
|------------|-------|---------|--------------|------------|
| FM         | 0.7763 | 0.3817  | 27.6         |            |
|            | 0.7771* | 0.3813* | 26.4*        | 3.98%      |
| DeepFM     | 0.7799 | 0.3796  | 46.2         |            |
|            | 0.7818* | 0.3785* | 43.8*        | 4.97%      |
| xDeepFM    | 0.7846 | 0.3771  | 85.8         |            |
|            | 0.7852* | 0.3767* | 81.0*        | 5.10%      |
| IPNN       | 0.7836 | 0.3782  | 58.8         |            |
|            | 0.7837 | 0.3777  | 52.2*        | 10.98%     |
| Wide&Deep  | 0.7776 | 0.3810  | 46.8         |            |
|            | 0.7781 | 0.3808  | 39.0*        | 17.02%     |
| DCN        | 0.7797 | 0.3797  | 58.2         |            |
|            | 0.7799 | 0.3797  | 54.0*        | 7.30%      |

Inference is based on a batch of test data (batchsize=2048). * indicates the statistically significant improvements (i.e., two-sided t-test with $p < 0.05$) over the original model.

3.5 Transferability Analysis (RQ2)

In this subsection, we will investigate the transferability of AutoField’s feature selection results. We apply its selected feature fields to six advanced deep recommendation models, namely Factorization Machine (FM) [35], DeepFM [11], xDeepFM [22], IPNN [34], Wide&Deep (WD) [5], DeepCrossNet (DCN) [39] on the Avazu dataset. In Table 2, the first line of each model is the performance with all possible feature fields, while the second line is that with only selected feature fields by AutoField. The AUC score, Logloss, and average inference time on a test batch are compared. It can be observed that:

- For all models, applying AutoField’s selection results not only improves the model performance but also reduces the inference time, which can enhance the online inference efficiency. This shows that the AutoField framework makes a general feature selection, which can also benefit other recommender systems.
• By applying AutoField’s selection results, we can observe that the recommendation performance improves more on the first three FM-based models, while the inference time reduces more on other models like IPNN, xDeepFM, Wide&Deep, DCN.

In summary, based on experimental results, we can conclude that the feature selection results of AutoField have a strong transferability to other deep recommendation models.

### Table 3: Ablation study on Avazu

| Model | AUC  | Logloss | Dropped Features Fields          |
|-------|------|---------|----------------------------------|
| AL1   | 0.7769 | 0.3816 | None                             |
| AL2   | 0.7615 | 0.3883 | Not Stable                       |
| AL3   | 0.7554 | 0.3920 | Not Stable                       |
| AutoField | 0.7773 | 0.3813 | [1, 5, 7, 8, 12, 13, 15, 16, 18, 19, 21] |

### 3.6 Ablation Study (RQ3)

In this subsection, we will study the contributions of important components of AutoField. We mainly analyze two components and design the following three variants:

- **AL-1**: In this variant, we use the argmax operation to make an automated selection rather than selecting \( K \) feature fields with the highest scores. In other words, AL-1 simply selects all feature fields with \( a_k^*>0.5,\forall n \in [1, N] \).
- **AL-2**: This variant conducts the feature selection using original Softmax instead of Gumbel-Softmax.
- **AL-3**: This variant involves both the above modifications in AL-1 and AL-2.

The results are shown in Table 3. We can observe that:

• Only the proposed AutoField framework can generate a stable result of feature selection.
• Making decisions by only argmax cannot achieve the goal of the feature selections. For example, AL-1 selects all the feature fields. The reason is that this variant overlooks the comparison with other feature fields.
• Original Softmax can lead to suboptimal model performance, such as AL-2. The reason might be that soft selection results in a gap between the search stage and the retraining stage.

With the ablation study above, we further validated the contribution of vital components of AutoField.

### 3.7 Parameter Analysis (RQ4)

In this subsection, we aim to investigate the importance of hyper-parameters. The most crucial one for AutoField is the number of selected feature fields, i.e., \( K \). We set \( K \) as \([8, 11, 14, 17]\) and fix all other hyper-parameters, then select the optimal feature fields via AutoField on Avazu dataset. The results are visualized in Figure 5. We can find that:

• The optimal \( K \) is 11, i.e., \( K^* = 11 \).
• With larger \( K \) like \( K \approx 14 \), AutoField will introduce some trivial features that hurt the model performance. When \( K \) becomes even larger like \( K \approx 17 \), AutoField could select feature fields that perform close to \( K = 22 \), i.e., the baseline All Fields in Table 1.

### 3.8 Case Study (RQ5)

In this subsection, we study whether the selected feature fields of AutoField are optimal from all possible choices on the MovieLens-1M dataset\(^6\). MovieLens-1M is a benchmark dataset offered by GroupLens Research. There are eight feature fields in this dataset. We convert the user ratings on movies (i.e., labels) to binary values. To be specific, user ratings above 3 are set as “like” (1), while lower ratings are set as “dislike” (0).

The result is visualized in Figure 6, where the x-axis is the number of selected feature fields \( K \). Blue points are the AUC scores of all possible feature selection choices, i.e., we enumerate all possible feature subsets. Orange points are the AUC scores of selected feature fields by AutoField with \( K = 4, 5, 6, 7 \). We could observe that the selected feature field of AutoField always achieves top performance among all possible solutions.

### 4 RELATED WORK

In this section, we will discuss the related work. The proposed AutoField is an AutoML-based feature selection model for recommendations. Thus, we will summarize the related work from two perspectives: feature selection techniques and AutoML models.

Feature selection is a popular topic that draws much attention from both academia and industry, which could help to remove redundant or irrelevant feature fields [6, 21, 26, 27, 48]. In general, feature selection methods could be divided into three categories [12], i.e., the Wrapper, Filter, and Embedded methods. Wrapper methods always elaborate a predictive model as a black-box to evaluate the performance of new subsets of feature fields [20]. As a result, this kind of solution is computationally intensive. Filter methods are data-oriented, which means the selection process is independent of subsequent models. Consequently, they require less computation.

\(^6\)https://grouplens.org/datasets/movielens/1m/
We propose a novel AutoML-based framework, AutoField, to im-
prove the performance of deep recommendation systems and speed up their inference progress by selecting informative feature fields. 

AutoField is capable of deciding the most effective subset of feature fields from the whole feature space. Specifically, we first provide a basic deep recommendation model to predict users’ preferences for items based on the input feature fields. Then, a controller is defined to decide which feature fields should be selected. Then, the feature selection module is introduced to bridge the deep recommendation model and controller. In detail, the feature selection module adapts the Gumbel–Softmax operation to simulate the hard selection in the search stage and applies the Softmax operation in the retraining stage. With optimized AutoField, we finally obtain the optimal subset of feature fields. We carried out substantial experiments to verify the effectiveness of AutoField on two benchmark datasets. The results indicate that AutoField could improve deep recommendation performance with outstanding transferability.

ACKNOWLEDGEMENT
This research was partially supported by Start-up Grant (No.9610565) for the New Faculty of the City University of Hong Kong and the CCF-Tencent Open Fund. Also, this research was partially supported by grants from the National Key Research and Development Program of China (Grant No. 2018YFC0117000), and the National Natural Science Foundation of China (Grant No.62072423).

A BASELINE SETTING
Principal Component Analysis (PCA) [40]. PCA is an eigen-
value decomposition-based multivariate analysis method for di-
mensionality deduction. It projects each data point onto only the
first few principal components to obtain lower-dimensional data
while preserving as much of the data’s variation as possible. We
choose the number of principal as few as possible, on the condition
that the selected components could explain more than 99.9% cov-
variance. With this assumption, 2 principal components for Avazu
and 8 principal components for Criteo are selected. To generate
the final results of feature selection, we add up the absolute value
of coefficients in all components and pick feature fields with the
highest value. The number of final selected feature fields is equal
to that of AutoField for a fair comparison.

Least Absolute Shrinkage and Selection Operator (LASSO)
[38]. LASSO is a regression analysis method for feature selection.
By introducing a $L_1$ norm, LASSO forces the sum of squared re-
gression coefficients to be less than a predefined value $\lambda$ and finally sets certain coefficients to zero, achieving the feature selection. It selects the feature fields automatically according to $\lambda$. To be specific, we use LASSO implementation in Python Library scikit-learn\(^7\). And for fairness, we set $\lambda = 0.1$ on Avazu and $\lambda = 0.15$ on Criteo. With these settings, similar to AutoField selections, LASSO drops 11 feature fields on Avazu and 5 feature fields on Criteo.

Gradient Boosting Technique [4]. Gradient Boosting is a ma-
chine learning technique widely used to ensemble models for better
performance. Gradient Boosting Regressor (GBR) is an ensem-
ble model based on regression models, while Gradient Boosting De-
cision Tree (GBDT) is based on decision trees. They both use an
impurity-based method to compute feature importance. For GBR,
like LASSO, it gives selection directly according to parameters. For
GBDT, like PCA, we select the most important $K$ feature fields, the
same number as AutoField.

IDARTS [18]. AutoField framework is encouraged by DARTS [24].
IDARTS is another differentiable Neural Architecture Search method,
which also could be applied to feature selection problems. The dif-
ference is that we use parameter pairs for each feature field that
control the feature selection and apply Gumbel-Softmax over each
pair to get the final score. IDARTS only assigns one parameter to
each feature field and applies softmax over all feature fields together
for scoring.

\(^7\)https://scikit-learn.org/stable/index.html
REFERENCES

[1] G. Anandalingam and Terry L. Friesz. 1992. Hierarchical optimization: An introduction. Annals of Operations Research 34, 1 (1992), 1–11.

[2] Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. IEEE transactions on pattern analysis and machine intelligence 35, 8 (2013), 1798–1828.

[3] James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. Journal of machine learning research 13, 2 (2012).

[4] Leo Breiman. 1994. Averting the edge. Technical Report. Technical Report 486, Statistics Department, University of California at Berkeley.

[5] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In Proceedings of the 1st workshop on deep learning for recommender systems. 7–10.

[6] Wei Fan, Kumpeng Liu, Hao Liu, Pengyang Wang, Yong Ge, and Yanjie Fu. 2020. AutoPS: Automated Feature selection via diversity-aware interactive reinforcement learning. In 2020 IEEE International Conference on Data Mining (ICDM). IEEE, 1008–1013.

[7] Valeria Fonti and Eduard Belitser. 2017. Feature selection using lasso. VU Amsterdam Research Paper in Business Analytics 30 (2017), 1–25.

[8] Jerome H Friedman. 2017. Arcing the edge. Journal of Machine Learning Research 18, 1 (2017), 2825–2839.

[9] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the thirteenth international conference on artificial intelligence and statistics. JMLR Workshop and Conference Proceedings, 249–256.

[10] Emili Julius Gumbel. 1954. Statistical theory of extreme values and some practical applications: a series of lectures. Vol. 33. U.S. Government Printing Office.

[11] Huafei Guo, Ruitong Wang, Yuning Yu, Zhenguo Li, and Xingqian He. 2017. DeepFM: a factorization-machine-based neural network for CTR prediction. arXiv preprint arXiv:1703.04247 (2017).

[12] Isabelle Guyon and André Elisseeff. 2003. An introduction to variable and feature selection. Journal of machine learning research 3, Mar (2003), 1157–1182.

[13] Mark Andrew Hall. 1999. Correlation-based feature selection for machine learning. (1999).

[14] Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval. 355–364.

[15] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In Proceedings of the 26th international conference on world wide web. 173–182.

[16] Geoffrey Hinton. 2012. A practical guide to training restricted Boltzmann machines. In Neural networks: Tricks of the trade. Springer, 599–619.

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

[18] Yufan Jiang, Chi Hu, Tong Xiao, Chunliang Zhang, and Jingbo Zhu. 2019. Improved differentiable architecture search for language modeling and named entity recognition. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP-IJCNLP). 3585–3590.

[19] Boaz Nadler and Ronald R Coifman. 2005. The prediction error in CLS and PLS: A comparison of two principal component regression methods. Journal of Machine Learning Research 6 (2005), 107–118.

[20] Yueting Zhuang. 2017. Social-aware movie recommendation via multimodal fusion of content and user social graph. Journal of Machine Learning Research 18, 28 (2017), 1–51. http://jmlr.org/papers/v18/16-330.html

[21] Xiaosa Zhao, Kunpeng Liu, Wei Fan, Lu Jiang, Xiaowei Zhao, Minghao Yin, and Yanjie Fu. 2020. Simplifying reinforced feature selection via restructured choice learning. In International Conference on Machine Learning. PMLR, 4095–4104.

[22] Boaz Nadler and Ronald R Coifman. 2005. The prediction error in CLS and PLS: A comparison of two principal component regression methods. Journal of Machine Learning Research 6 (2005), 107–118.

[23] Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. IEEE transactions on pattern analysis and machine intelligence 35, 8 (2013), 1798–1828.

[24] Haochen Liu, Xiangyu Zhao, Chong Wang, Xiaobing Liu, and Jiliang Tang. 2020. Efficient neural architecture search via parameters sharing. In International Conference on Machine Learning. PMLR, 291–302.

[25] Hieu Pham, melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. 2018. Efficient neural architecture search via parameter sharing. In Proceedings of the 4th Conference on Artificial Intelligence and Statistics. PMLR, 949–957.

[26] J Ben Schuler, Dan Frankowski, Jon Herlocker, and Shladi Sen. 2007. Collaborative filtering recommender systems. In The adaptive web. Springer, 291–314.

[27] Anastasis Kratsios and Cody Hyndman. 2021. NEU: A Meta-Algorithm for Universal UAP-Invariant Feature Representation. Journal of Machine Learning Research 22, 92 (2021), 1–51. http://jmlr.org/papers/v22/v18-805.html

[28] Jitao Liu, Xiaolun Zhu, Fuwen Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. xdeepfm. Combining explicit and implicit feature interactions for recommender systems. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 1754–1763.

[29] Haochen Liu, Chenxu Zhu, Guolin Li, Weinan Zhang, Jincui Lai, Ruiming Tang, Xiaojing He, Zhenguo Li, and Yong Yu. 2020. Autofm: Automatic feature interaction selection in factorization models for click-through rate prediction. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2636–2645.

[30] Hanxiao Liu, Yuanfei Luo, Mengshuo Wang, Hao Zhou, Quanzhong Yao, Wei-Wei Tu, Yuqiang Chen, Wenyuan Dai, and Quanquan Gao. 2021. AutoCross: Automated feature selection via restructured choice strategy of single agent. In 2020 IEEE International Conference on Data Mining (ICDM). IEEE, 871–880.

[31] Hanxiao Liu, Yuanfei Luo, Mengshuo Wang, Hao Zhou, Quanzhong Yao, Wei-Wei Tu, Yuqiang Chen, Wenyuan Dai, and Quanquan Gao. 2021. AutoCross: Automated feature selection via restructured choice strategy of single agent. In 2020 IEEE International Conference on Data Mining (ICDM). IEEE, 871–880.