An Embedding Learning Framework for Numerical Features in CTR Prediction

Huifeng Guo\textsuperscript{1}\textsuperscript{*}, Bo Chen\textsuperscript{1}, Ruiming Tang\textsuperscript{1}, Weinan Zhang\textsuperscript{2}, Zhenguo Li\textsuperscript{1}, Xiuqiang He\textsuperscript{1}

\textsuperscript{1}Huawei Noah's Ark Lab \textsuperscript{2}Shanghai Jiao Tong University

\{huifeng.guo, chenbo116, tangruiming, li.zhenguo, hexiuqiang1\}@huawei.com; wnzhang@sjtu.edu.cn

ABSTRACT

Click-Through Rate (CTR) prediction is critical for industrial recommender systems, where most deep CTR models follow an Embedding & Feature Interaction paradigm. However, the majority of methods focus on designing network architectures to better capture feature interactions while the feature embedding, especially for numerical features, has been overlooked. Existing approaches for numerical features are difficult to capture informative knowledge because of the low capacity or hard discretization based on the offline expertise feature engineering. In this paper, we propose a novel embedding learning framework for numerical features in CTR prediction (AutoDis) with high model capacity, end-to-end training and unique representation properties preserved. AutoDis consists of three core components: meta-embeddings, automatic discretization and aggregation. Specifically, we propose meta-embeddings for each numerical field to learn global knowledge from the perspective of a field with a managable number of parameters. Then the differentiable automatic discretization performs soft discretization and captures the correlations between the numerical features and meta-embeddings. Finally, distinctive and informative embeddings are learned via an aggregation function. Comprehensive experiments on two public and one industrial datasets are conducted to validate the effectiveness of AutoDis. Moreover, AutoDis has been deployed onto a mainstream advertising platform, where online A/B test demonstrates the improvement over the base model by 2.1% and 2.7% in terms of CTR and eCPM, respectively. In addition, the code of our framework is publicly available in MindSpore\textsuperscript{1}.

CCS CONCEPTS

- Information systems → Information retrieval. Recommender systems;
- Computing methodologies → Machine learning.

\textsuperscript{1}https://gitee.com/mindspore/mindspore/tree/master/model_zoo/research/recommend/autodis

\textsuperscript{*}Co-first authors with equal contributions. Huifeng Guo and Ruiming Tang are the corresponding authors.

1 INTRODUCTION

To alleviate the problem of information explosion, recommender systems are widely deployed to provide personalized information filtering in online information services, such as web search, commodity recommendation and online advertising. CTR prediction is crucial in recommender systems, which is to estimate the probability that a user will click on a recommended item under a specific context, so that recommendation decisions can be made based on the predicted CTR values [20, 33]. Because of the superior performance of representation learning in computer vision and natural language processing, deep learning techniques attract the attention of recommendation community. Therefore, plenty of industrial companies propose various deep CTR models and deploy them in their commercial systems, such as Wide & Deep [2] in Google Play, DeepFM [7] in Huawei AppGallery and DIN [33] in Taobao.

As shown in Figure 1, most of the existing deep CTR models follow an Embedding & Feature Interaction (FI) paradigm. Due to the significant importance for feature interaction in CTR prediction, the majority works focus on designing network architectures in FI module to better capture explicit or implicit feature interactions, e.g., wide component in Wide & Deep [2], CrossNet in DCN [28], and attention mechanism in DIN [33]. Although not well studied in the literature, Embedding module is also a key factor to deep CTR models for two reasons: (1) Embedding module is the cornerstone of the subsequent FI module and directly affects the effectiveness of FI module [30]; (2) The number of parameters in deep CTR models is heavily concentrated in Embedding module, which naturally has high impact on the prediction performance [11]. However, the Embedding module is overlooked by the research community, which motivates us to conduct in-depth research.

The Embedding module works in a look-up table manner generally and maps each feature of the input data in categorical fields\textsuperscript{2}, to the latent embedding space with learnable parameters [7]. Considering an example instance (Gender=Male, Day=Tuesday, Height=175.6, Weight=89.1, Age=33), the Embedding module can convert the above field values to a 2D vector space, where each dimension represents a specific feature value. As a result, the Embedding module captures the field-to-field interactions and learns the latent representations of each feature. This process is essential for accurately predicting the user’s behavior, such as click-through rate (CTR) prediction and other related applications.
Age=18), for categorical field Gender, features are mapped to embeddings trivially by assigning a unique embedding to each individual feature, which is widely used in existing works [7, 24]. Unfortunately, such categorization strategy cannot be utilized to handle numerical features, as there may be infinitely many feature values in a numerical field (e.g., Height).

In practical, existing representation approaches for numerical feature can be summarized into three categories (red dashed box in Figure 1): (1) No Embedding: using the original feature values or its transformation directly without learning embeddings [2, 3, 21]; (2) Field Embedding: learning a single field embedding for each numerical field [7, 26]; (3) Discretization: transforming numerical features to categorical features through various heuristic discretization strategies and assigning embedding like categorization strategy [24]. However, the former two categories may lead to poor performance due to the low capacity of representations. The last category is also sub-optimal because such heuristic-based discretization rules cannot be optimized with the ultimate goal of CTR models. Besides, hard discretization-based methods suffer from SBD (Similar value But Dis-similar embedding) and DBS (Dis-similar value But Same embedding) problems whose details will be discussed in Session 2.

To address the limitations of existing approaches, we propose an automatic end-to-end embedding learning framework for numerical features based on soft discretization (AutoDis). AutoDis consists of three core modules: meta-embeddings, automatic discretization and aggregation to enable high model capacity, end-to-end training and unique representation properties. Specifically, we elaborately design a set of meta-embeddings for each numerical feature and field-specific meta-embeddings. Finally, an aggregation function is performed to learn unique Continuous-But-Different representations. To the best of our knowledge, AutoDis is the first end-to-end soft discretization framework for numerical feature embedding that can be optimized jointly with the ultimate goal of deep CTR models. We summarize the main contributions as follows.

- We propose AutoDis, a pluggable embedding learning framework for numerical features, which has high model capacity and is able to generate unique representations with a manageable number of parameters in an end-to-end manner.

- In AutoDis, we design meta-embeddings for each numerical field to learn global shared knowledge. Besides, a differentiable automatic discretization is used to capture the correlations between the numerical features and meta-embeddings, while an aggregation process is performed to learn a unique Continuous-But-Different representation for each feature.

- Comprehensive experiments are conducted on two public and an industrial datasets to demonstrate the superiority of AutoDis over the existing representation methods for numerical features. Moreover, AutoDis works compatibly with various popular deep CTR models, by improving their recommendation performance significantly. Online A/B test in a mainstream advertising platform shows that AutoDis improves the commercial baseline by 2.1% and 2.7% in terms of CTR and eCPM.

2 PRELIMINARY

In this section, we delve into the feature embedding learning procedure in CTR prediction. Suppose a dataset for training CTR models consists of \( Q \) instances \((x, y)\), where \( y \) is label and \( x \) is a multi-field data record including \( M \) categorical fields and \( N \) numerical fields:

\[
x = [x_1, x_2, \ldots, x_M; x_1, x_2, \ldots, x_N],
\]

where \( x_1 \) is the one-hot vector of the feature value in the \( i \)-th categorical field and \( x_j \) is the scalar value of the \( j \)-th numerical field.

For the \( i \)-th categorical field, the feature embedding can be obtained by embedding look-up operation:

\[
e_i = E_i \cdot x_i,
\]

where \( E_i \in \mathbb{R}^{d \times d} \) is the embedding matrix for \( i \)-th field, \( v_i \) and \( d \) is the vocabulary size and embedding size. Therefore, the representations for categorical fields are constructed as \([e_1, e_2, \ldots, e_M] \in \mathbb{R}^{M \times d} \). For numerical fields, existing representation approaches can be summarized into three categories: No Embedding, Field Embedding and Discretization.

2.1 Category 1: No Embedding

This category uses the original values or the transformations directly without learning embedding. For instance, Wide & Deep [2] of Google Play and DMT [6] of JD.com use the original and normalized numerical features respectively. Besides, YouTube DNN [3] utilizes several transformations (i.e., square and square root) of
normalized feature value \( \hat{x}_j \):
\[
e_{\text{infinitube}} = \{ x_1, x_1, \sqrt{x_1}, x_2, \sqrt{x_2}, \ldots, x_N, \sqrt{x_N} \},
\]
where \( e_{\text{infinitube}} \in \mathbb{R}^N \). In DLRM \[21\] of Facebook, they use a multi-layer perception to model all numerical features:
\[
e_{\text{DLRM}} = \{ \text{DNN}(x_1, x_2, \ldots, x_N) \},
\]
where the structure of DNN is 512-256-64 and \( e_{\text{DLRM}} \in \mathbb{R}^d \). Intuitively, these No Embedding methods are difficult to capture informative knowledge of numerical fields due to their low capacity.

2.2 Category 2: Field Embedding
The commonly used method in academia is Field Embedding \[7, 26\], where all the numerical features in the same field share a uniform field embedding and then multiply this field embedding with their feature values:
\[
e_{\text{FE}} = \{ x_1 \cdot e_1, x_2 \cdot e_2, \ldots, x_N \cdot e_N \},
\]
where \( e_j \in \mathbb{R}^d \) is the uniform embedding vector for the \( j \)-th numerical feature field and \( e_{\text{FE}} \in \mathbb{R}^{Nd} \). However, the representation capacity of Field Embedding is limited due to the single shared field-specific embedding as well as the linear scaling relationship between different intra-field features.

2.3 Category 3: Discretization
A popular method in industrial recommender systems to handle numerical features is Discretization, i.e., transforming numerical features to categorical features. For a feature \( x_j \) in the \( j \)-th numerical field, the feature embedding \( e_j \) can be obtained by a two-stage operation: discretization and embedding look-up.
\[
e_j = E_j \cdot d_j(x_j),
\]
where \( E_j \in \mathbb{R}^{H_j \times d} \) is the embedding matrix for \( j \)-th field, \( H_j \) is the number of buckets after discretization. Function \( d_j(\cdot) \) is the manually-designed discretization method for the \( j \)-th numerical field, which project each feature value into one of the \( H_j \) buckets. Specifically, there are three widely-used discretization approaches.
1. EDD/EFD (Equal Distance/Frequency Discretization). EDD partitions the feature values into \( H_j \) equal-width buckets. Suppose the range of features of the \( j \)-th field is \([x_{j \min}^\text{max}, x_{j \max}^\text{max}]\), the interval width is defined as \( w_j = (x_{j \max}^\text{max} - x_{j \min}^\text{max}) / H_j \). Accordingly, the discretized result \( \hat{x}_j \) is obtained via EDD discretization function \( d_{j\text{EDD}}(\cdot) \):
\[
\hat{x}_j = d_{j\text{EDD}}(x_j) = \text{floor}((x_j - x_{j \min}^\text{min}) / w_j).
\]
Similarly, EFD divides the range \([x_{j \min}^\text{min}, x_{j \min}^\text{max}]\) to several buckets such that each bucket contains equal number of features.
2. LD (Logarithm Discretization). The champion \[12\] of Criteo advertiser prediction in Kaggle Competition\(^3\) utilizes the logarithm and floor operation to transform the numerical features to categorical form. The bucket \( \hat{x}_j \) after discretized is obtained by \( d_{j\text{LD}}(\cdot) \):
\[
\hat{x}_j = d_{j\text{LD}}(x_j) = \text{floor}(\log(x_j)^2).
\]

Figure 2: Limitations of existing discretization approaches: (1) TPP (Two-Phase Problem); (2) SBD (Similar value But Dis-similar embedding); DBS (Dis-similar value But Same embedding).

3 METHODOLOGY
To address the limitations of the existing approaches, we propose AutoDis, which automatically learns distinctive embeddings for numerical features in an end-to-end manner. As shown in Figure 3,

\(^3\)http://https://www.kaggle.com/c/criteo-display-ad-challenge/discussion/10555
A naive solution is to treat numerical features as categorical features. However, this method has serious defects, such as explosively huge parameters and insufficient training for low-frequency features, which is unaffordable and inadvisable in practice. On the contrary, Field Embedding method shares a single embedding among all the feature values within a field for saving parameters, degrading the model performance due to the low capacity. To balance the model capacity and complexity, for the j-th field, we design a set of meta-embeddings $\mathbf{ME}_j \in \mathbb{R}^{H_j \times d_j}$ shared by features within this field, where $H_j$ is the number of meta-embeddings. Meta-embeddings are conductive to learn the intra-field shared global knowledge. Each meta-embedding can be viewed as a sub-space in the latent space for boosting expressiveness and capacity. By combining these meta-embeddings, the learned embedding is more informative than that of the Field Embedding method, so that the high model capacity can be well preserved. Moreover, the parameters needed are determined by $H_j$. Consequently, the model complexity is highly controllable, making our method scalable.

### 3.3 Automatic Discretization

To capture complicated correlations between a numerical feature value and the designed meta-embeddings, we meticulously propose a differentiable automatic discretization module $d_j^{\text{Auto}}(\cdot)$. Thanks to $d_j^{\text{Auto}}(\cdot)$, each numerical feature in the $j$-th field is discretized with $H_j$ buckets where each bucket embedding corresponds to a meta-embedding mentioned-above.

Specifically, a two-layer neural network with skip-connection is leveraged to discretize the feature value $x_j$ to $H_j$ buckets by:

$$\mathbf{h}_j = \text{LeakyReLU}(\mathbf{w}_j x_j + \mathbf{a} \mathbf{h}_j),$$

$$\tilde{x}_j = \mathbf{W}_j \mathbf{h}_j + \alpha \mathbf{h}_j,$$

where $\mathbf{w}_j \in \mathbb{R}^{1 \times H_j}$ and $\mathbf{W}_j \in \mathbb{R}^{H_j \times H_j}$ are the learnable parameters of the automatic discretization network for the $j$-th numerical feature field, activation function is LeakyReLU [8] and $\alpha$ is the control factor of skip-connection. The projecting result is $\tilde{x}_j = \{\tilde{x}_{j1}, \tilde{x}_{j2}, \ldots, \tilde{x}_{jH_j}\}$, where $\tilde{x}_{jh}$ is the projection output of feature value $x_j$ on the $h$-th bucket. Finally, the correlations between feature value $x_j$ and each of the meta-embeddings $\mathbf{ME}_j$ is normalized via Softmax function, denoted as:

$$\tilde{x}_j^h = \frac{\tau^{\tilde{x}_{jh}}}{\sum_{h=1}^{H_j} \tau^{\tilde{x}_{jh}}},$$

where $\tau$ is the temperature coefficient to control discretization distribution. Therefore, the discretized result $\tilde{x}_j \in \mathbb{R}^{H_j}$ is obtained through the automatic discretization function $d_j^{\text{Auto}}(\cdot)$:

$$\tilde{x}_j = d_j^{\text{Auto}}(x_j) = \{\tilde{x}_{j1}, \ldots, \tilde{x}_{jH_j}\}. \quad (13)$$

The discretized result $\tilde{x}_j$ is a vector where each element $\tilde{x}_{jh}$ denotes the probability that feature $x_j$ is discretized to the $h$-th bucket, which represents the correlation between feature value $x_j$ and the $h$-th meta-embedding (i.e., bucket embedding). This discretization method can be understood as soft discretization. Compared with

![Figure 4: The architecture of AutoDis framework.](image-url)
the hard discretization mentioned in Section 2.3, soft discretization does not discretize feature values into a single determinate bucket, so that the SBD and DBS issues can be well solved. Moreover, the differentiable soft discretization enables our AutoDis to achieve end-to-end training and optimize with the ultimate goal.

It is worth noting that when temperature coefficient in $\tau \to \infty$, the discretization distribution in Eq.(13) tends to be uniform and when $\tau \to 0$, the distribution tends to be one-hot. Therefore, temperature coefficient $\tau$ plays a significant role in the automatic discretization distribution. Besides, the feature distribution is diverse over different fields, so it is of great necessity to learn different $\tau$ for different features. Specifically, we propose a temperature co-efficient adaptive network, which takes both global field statistics feature and local input feature into consideration, shown as:

$$\tau_{x_j} = \text{Sigmoid}(W_j^2 \text{LeakyReLU}(W_j^1[n_j|x_j])),$$ (14)

where $n_j$ is the global statistics feature vector of the $j$-th field including sampled Cumulative Distribution Function (CDF) and mean value. $x_j$ is the local input feature, $W_j^1$ and $W_j^2$ are the weight parameters. To guide model training, the output range of $\tau_{x_j}$ is rescale from $(0, 1)$ to $(\tau - e, \tau + e)$, where $\tau$ is a global hyperparameter.

### 3.4 Aggregation Function

After getting the correlations between a feature value and the meta-embeddings, the embeddings can be generated by applying aggregation operation $f(\cdot)$ over the meta-embeddings with the correlations as the selection probabilities. The following aggregation functions $f(\cdot)$ are considered:

- **Max-Pooling** selects the most relevant meta-embedding with highest probability $\hat{x}_{jk}$:

$$e_j = \text{ME}^k_j, \text{where} \; k = \arg \max_{h \in \{1, \ldots, H_j\}} \hat{x}_{jk},$$ (15)

where $k$ is the index of the most relevant meta-embedding with highest probability $\hat{x}_{jk}$ and $\text{ME}^k_j$ is the $k$-th meta-embedding in $\text{ME}_j$. However, this hard selection strategy degenerates AutoDis to a hard discretization method, resulting in aforementioned SBD and DBS problems.

- **Top-K-Sum** sums up top-K meta-embeddings with highest correlation $\hat{x}_{jk}$:

$$e_j = \sum_{l=1}^{K} \text{ME}^k_j, \text{where} \; k = \arg \max_{h \in \{1, \ldots, H_j\}} \hat{x}_{jk},$$ (16)

where $k_l$ is the index of the meta-embedding with the $l$-th ($l \in [1, K]$) largest $\hat{x}_{jk}$, $\text{ME}^k_j$ is the $k$-th meta-embedding in $\text{ME}_j$. However, Top-K-Sum approach suffers from two limitations: (1) It cannot fundamentally solve DBS problem although the number of possible generated embeddings increases from $H_j$ to $C^K_{H_j}$ in comparison with Max-Pooling; (2) The learned embedding $e_j$ does not consider the correlation values.

- To take advantage of the entire set of meta-embeddings and their correlations to the feature value sufficiently, a Weighted-Average aggregation function is proposed as $e_j = \sum_{h \in \{1, \ldots, H_j\}} \hat{x}_{jh} \text{ME}^h_j$. With this weighted aggregation strategy, relevant meta-embeddings contribute more to providing an informative embedding while irrelevant meta-embeddings are largely neglected.

### 3.5 Training

AutoDis is trained jointly with the ultimate goal of a concrete deep CTR model in an end-to-end manner. The loss function is the widely-used LogLoss with a regularization term:

$$\mathcal{L} = -\frac{1}{Q} \sum_{i=1}^{Q} y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) + \lambda \|\Theta\|_2, \quad (17)$$

where $y_i$ and $\hat{y}_i$ are the ground truth label and estimated value of the $i$-th instance, respectively. $Q$ is the total number of training instances and $\lambda$ is the $L_2$ regularization weight. $\Theta = \{\Theta_{\text{CatEmb}}^{\text{Emb}}, \Theta_{\text{AutoDis}}^{\text{Emb}}, \Theta_{\text{CTR}}\}$ are the parameters of feature embeddings in categorical fields, parameters of meta-embeddings and automatic discretization (i.e., Eq.(11-14)) in AutoDis, as well as deep CTR model parameters.

To stabilize the training process, we adopt the feature normalization technique to scale numerical feature values into $[0, 1]$ in the data pre-processing stage. A numerical feature value $x_j$ of the $j$-th numerical field is normalized as $x_j \leftarrow \frac{x_j - x_{j}^{\min}}{x_{j}^{\max} - x_{j}^{\min}}$.

### 4 EXPERIMENTS

#### 4.1 Experimental Setting

##### 4.1.1 Dataset and Evaluation Protocols

To evaluate the effectiveness of our proposed numerical feature embedding learning framework AutoDis, we conduct extensive experiments on two popular benchmarks: Criteo, AutoML and one industrial dataset. Table 2 summarizes the statistics of all the three datasets.

| Dataset  | #Num field | #Cats field | #Instances $\times 10^6$ | Positive Ratio |
|----------|------------|-------------|--------------------------|----------------|
| Criteo   | 13         | 26          | 45.8                     | 25.6%          |
| AutoML   | 23         | 51          | 4.69                     | 5.8%           |
| Industrial | 41     | 44          | 87.5                     | 3.3%           |

Besides, this strategy ensures that each feature learn unique representation, and meanwhile, the learned embeddings are Continuous-But-Different, meaning that the closer the feature values, the more similar the embeddings are.

#### 4.5 Training

AutoDis is trained with the ultimate goal of a concrete deep CTR model in an end-to-end manner. The loss function is the widely-used LogLoss with a regularization term:

$$\mathcal{L} = -\frac{1}{Q} \sum_{i=1}^{Q} y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) + \lambda \|\Theta\|_2, \quad (17)$$

where $y_i$ and $\hat{y}_i$ are the ground truth label and estimated value of the $i$-th instance, respectively. $Q$ is the total number of training instances and $\lambda$ is the $L_2$ regularization weight. $\Theta = \{\Theta_{\text{CatEmb}}^{\text{Emb}}, \Theta_{\text{AutoDis}}^{\text{Emb}}, \Theta_{\text{CTR}}\}$ are the parameters of feature embeddings in categorical fields, parameters of meta-embeddings and automatic discretization (i.e., Eq.(11-14)) in AutoDis, as well as deep CTR model parameters.

To stabilize the training process, we adopt the feature normalization technique to scale numerical feature values into $[0, 1]$ in the data pre-processing stage. A numerical feature value $x_j$ of the $j$-th numerical field is normalized as $x_j \leftarrow \frac{x_j - x_{j}^{\min}}{x_{j}^{\max} - x_{j}^{\min}}$.

### 4 EXPERIMENTS

#### 4.1 Experimental Setting

##### 4.1.1 Dataset and Evaluation Protocols

To evaluate the effectiveness of our proposed numerical feature embedding learning framework AutoDis, we conduct extensive experiments on two popular benchmarks: Criteo, AutoML and one industrial dataset. Table 2 summarizes the statistics of all the three datasets.

| Dataset  | #Num field | #Cats field | #Instances $\times 10^6$ | Positive Ratio |
|----------|------------|-------------|--------------------------|----------------|
| Criteo   | 13         | 26          | 45.8                     | 25.6%          |
| AutoML   | 23         | 51          | 4.69                     | 5.8%           |
| Industrial | 41     | 44          | 87.5                     | 3.3%           |

Besides, this strategy ensures that each feature learn unique representation, and meanwhile, the learned embeddings are Continuous-But-Different, meaning that the closer the feature values, the more similar the embeddings are.

#### 3.5 Training

AutoDis is trained jointly with the ultimate goal of a concrete deep CTR model in an end-to-end manner. The loss function is the widely-used LogLoss with a regularization term:

$$\mathcal{L} = -\frac{1}{Q} \sum_{i=1}^{Q} y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) + \lambda \|\Theta\|_2, \quad (17)$$

where $y_i$ and $\hat{y}_i$ are the ground truth label and estimated value of the $i$-th instance, respectively. $Q$ is the total number of training instances and $\lambda$ is the $L_2$ regularization weight. $\Theta = \{\Theta_{\text{CatEmb}}^{\text{Emb}}, \Theta_{\text{AutoDis}}^{\text{Emb}}, \Theta_{\text{CTR}}\}$ are the parameters of feature embeddings in categorical fields, parameters of meta-embeddings and automatic discretization (i.e., Eq.(11-14)) in AutoDis, as well as deep CTR model parameters.

To stabilize the training process, we adopt the feature normalization technique to scale numerical feature values into $[0, 1]$ in the data pre-processing stage. A numerical feature value $x_j$ of the $j$-th numerical field is normalized as $x_j \leftarrow \frac{x_j - x_{j}^{\min}}{x_{j}^{\max} - x_{j}^{\min}}$.
4.1.2 Baselines and Hyper-parameters Settings. To demonstrate the effectiveness of the proposed AutoDis, we compare AutoDis with three categories representation learning methods for numerical features: (1) No Embedding (YouTube [3]) DLRM [21]); (2) Field Embedding (FE [7]); (3) Discretization (EDD [24], LD as well as TD, namely DeepGBM [14]).

Moreover, to validate the compatibility of AutoDis framework with various embedding-based deep CTR models, we apply AutoDis to six representative models: FNN [30], Wide & Deep [2], DeepFM [7], DCN [28], IPNN [24] and xDeepFM [16].

Our implementation is publicly available in MindSpore. We optimize all the models with mini-batch Adam [15], where the learning rate is searched from {10e-6, 5e-5, ..., 10e-2}. Besides, the embedding size is set to 80 and 70 in Criteo and AutoML dataset. Specifically, the hidden layers in deep CTR models are fix to 1024-512-256-128 by default and the explicit feature interactions in DCN and xDeepFM (namely, CrossNet and CIN) are set to 3 layers. The weight of $L_2$ regularization is tuned in {10e-6, 5e-5, ..., 10e-3}. For AutoDis, the number of meta-embeddings in each numerical field is 20 and 40 in Criteo and AutoML. The skip-connection factor is searched in {0, 0.1, 0.1, 1} and the number of neurons in temperature coefficient adaptive network is set to 64.

4.2 Overall Performance

In this section, we first compare the performance of different representation learning methods for numerical features. Then, we evaluate the compatibility of AutoDis by showing the improvement over six popular deep CTR models.

4.2.1 Compared with Other Representation Learning Methods. We perform different numerical feature representation learning methods on three datasets and choose DeepFM as the deep CTR model. The performance comparison is presented in Table 3, from which, we have the following observations:

- The baselines perform inconsistently in different datasets due to discrepant characteristics of numerical features. Comparatively speaking, AutoDis outperforms all the baselines on all the datasets consistently by a significant margin, demonstrating the superiority and robustness to different numerical features.
- No Embedding and Field Embedding methods perform worse than Discretization and AutoDis. No Embedding methods achieve better results on industrial dataset than on public datasets, while Field Embedding method performs the opposite. These two category methods suffer from low capacity and limited expressiveness. On the one hand, No Embedding in YouTube is parameter-free and DLRM leverages a DNN to obtain a mixed representation of all the numerical fields, ignoring the inter-field distinction, which limits the performance significantly. On the other hand, for Field Embedding, all the feature values in the same field share a single embedding and the learned embeddings are the linear scaling of the shared embedding with the feature values. Such sharing and linear scaling strategy limits the performance fatally.
- In comparison with the three existing representative Discretization methods that transform the numerical features into categorical form, AutoDis improves AUC over the best baseline by 0.17%, 0.23%, 0.22%, respectively. Although Discretization methods achieve better performance comparison, the previous two categories, they cannot be optimized with the CTR models, and also suffer from SBD and DBS problems. The heuristic rule-based hard discretization strategies result in step-wise “unsmooth” embeddings. On the contrary, AutoDis overcomes these disadvantages by sharing a set of meta-embeddings for each numerical field and leveraging differentiable soft discretization and aggregation strategy to learn informative Continuous-But-Different representations.

4.2.2 Compatibility with Different CTR Models. AutoDis is a universal framework and can be regarded as plug-in component for improving various deep CTR models performance. To demonstrate its compatibility, in this section, we conduct extensive experiments by applying AutoDis on a series of popular models. From Table 4 we can observe that, compared with the Field Embedding representation method, AutoDis framework significantly improves the prediction performance of these models. The numerical feature discretization and embedding learning procedure is optimized with the ultimate goal of these CTR models, so that the informative representations can be obtained and the performance can be improved. In fact, a small improvement in AUC is likely to yield a significant online CTR increase and huge profits [2, 24].

4.3 Deployment & Online A/B Testing

Online experiments were conducted in a mainstream advertising platform for a weak to verify the superior performance of AutoDis, where millions of daily active users interact with ads and generate tens of millions of user log events. In the online serving system,
thousands of candidate ads that are most popular or relevant to users are retrieved from the universal ads pool and then ranked by a highly-optimized deep CTR model before presenting to users.

To guarantee user experience, the overall latency of the above-mentioned candidate recall and ranking is required to be within a few milliseconds. The commercial model is deployed in a cluster, where each node is with 48 core Intel Xeon CPU E5-2670 (2.30GHZ), 400GB RAM as well as 2 NVIDIA TESLA V100 GPU cards. For the control group, the numerical features are discretized via a variety of hybrid manually-designed rules (e.g., EDD, TD, etc.). The experiment group chooses AutoDis to discretize and learn embeddings automatically for all the numerical features. Deploying AutoDis to an existing CTR model is convenient and requires nearly no engineering work in the online serving system.

AutoDis achieves an offline AUC 0.2% improvement, and upgrades the online CTR and eCPM by 2.1% and 2.7% relative to the control group (statistically significant), which brings enormous commercial profits. Moreover, with AutoDis integrated, the existing numerical features no longer require any discretization rules. Furthermore, introducing more numerical features in the future will be more efficient, without exploring any handcrafted rules.

4.4 Embedding Analysis

To understand the Continuous-But-Different embeddings learned via AutoDis more deeply with insights, we do further investigation in macro-analysis of embeddings and micro-analysis of soft discretization, respectively.

4.4.1 Macro-Analysis of Embeddings. Figure 5 provides a visualization of the embeddings derived from DeepFM-AutoDis and DeepFM-EDD in the 3-rd numerical field of the Criteo dataset. We random select 250 embeddings and project them onto a two dimensional space using t-SNE [27]. Nodes with similar colors have similar values. We can observe that AutoDis learns a unique embedding for each feature. Moreover, similar numerical features (with similar colors) are represented by close-correlated embeddings (with similar positions in two-dimensional space), which elaborates the Continuous-But-Different property of the embeddings. However, EDD learns identical embedding for all the features within a bucket and completely different embeddings across different buckets, which leads to step-wise "unsmooth" embeddings and consequently results in inferior task performance.

4.4.2 Micro-Analysis of Soft Discretization. In this section, we perform some micro-level analysis by investigating the Softmax distribution during the soft discretization process from DeepFM-AutoDis. We select an adjacent feature-pair (feature value “1” and “2”) and a distant feature-pair (feature value “1” and “10”) from the 8-th numerical field of the Criteo dataset, and then visualize their discretization distributions in Figure 6. We can observe that, the adjacent feature-pair have similar Softmax distributions while distant feature-pair have diverse distributions. This characteristic is conducive to ensure that similar feature values can learn similar embeddings by AutoDis, so that the embedding continuity can be preserved.

4.5 Numerical Fields Analysis

To evaluate the effect of DeepFM-AutoDis on each numerical field, we select all the 26 categorical fields as the basic features and add one of the 13 numerical fields at a time cumulatively in the Criteo dataset. Figure 7 plots the prediction performance w.r.t adding numerical fields in the original order of the dataset and in a random order. From Figure 7, we have the following observations: (1) AutoDis improves the performance even if there is only a single numerical field; (2) AutoDis has a cumulative improvement on multiple numerical fields; (3) Compared with the existing methods, the performance improvement achieved by AutoDis is more significant and stable.

4.6 Model Complexity

To quantitatively analyze the space and time complexity of our proposed AutoDis, the model parameters and batch inference time with the EDD discretization method on DeepFM are compared. Table 5 reports the results on Criteo dataset. We can observe that, compared with EDD, the increased amount of model parameters by AutoDis are negligible. As online recommendation services usually have high requirements on latency, the computational cost during inference is important. We can find that AutoDis achieves
acceptable inference time compared to EDD, demonstrating its computational efficiency. Furthermore, AutoDis is an end-to-end framework, indicating that there is no extra efficiency overhead. However, other Discretization methods need additional overhead, e.g., manual design and data preprocessing in EDD; another GBDT model training in TD.

4.7 Ablation Study and Hyper-Parameters Sensitivity

4.7.1 Aggregation Strategy. To verify the effect of different aggregation strategies in Section 3.4, we perform ablation study. Figure 8 summarizes the prediction performance of DeepFM-AutoDis with Max-Pooling, Top-K-Sum and Weighted-Average aggregation strategies. It can be concluded that Weighted-Average strategy achieves the best performance. The reason is that Weighted-Average makes full use of the meta-embeddings and their corresponding correlations, and overcomes the DBS and SBD problems completely compared with other strategies.

4.7.2 Temperature Coefficient. To demonstrate the effectiveness of temperature coefficient adaptive network mentioned in Section 3.3 Automatic Discretization for generating feature-specific $r_\tau$, we compare the prediction results with global temperature coefficient $\tau$. The global temperature coefficients are searched and summarized in Figure 9. We find that the optimal global temperature is around 1e-5 and 4e-3 on Criteo and AutoML datasets, respectively. However, our temperature coefficient adaptive network achieves best results (red dashed line) because it can adjust temperature coefficient adaptively for different feature based on the global field statistics feature and local input feature, obtaining more appropriate discretization distribution.

4.7.3 Number of Meta-Embeddings. We study the number of meta-embedding in this section and vary it in the range of $\{10, 20, \ldots, 100\}$. The performance comparison is presented in Figure 10. It can be observed that increasing the number of meta-embeddings contributes to improving the performance substantially at the beginning. The reason lies in involving richer information in meta-embeddings. However, using too many meta-embeddings not only increases the computational burden, but also frustrates the performance. Considering both prediction effect and training efficiency, we set the number to 20 and 40 in Criteo and AutoML datasets, respectively.

5 RELATED WORK

As presented in Figure 1, most existing deep CTR models [2, 7, 22, 29, 33] follow an Embedding & Feature Interaction (FI) paradigm. Embedding module maps input features to low-dimensional embeddings. Based on such learnable embeddings, FI module models implicit/explicit low/high-order feature interactions by learning nonlinear relationship among features. Below, we will give a brief introduction of these modules in deep CTR models.

5.1 Embedding

As the cornerstone of the deep CTR models, Embedding module has significant impact on model performance because it takes up majority of the model parameters. Since the approaches for numerical features have been discussed in detail in Section 2, we here give a brief introduction about research based on embedding look-up in recommendation domain. Existing researches mainly focus on designing adaptable embedding algorithms by allocating variable-length embeddings or multi-embeddings to different features, such as Mixed-dimension [4], NIS [11] and AutoEmb [31]. Another research line delves into the embedding compression [13] to reduce the memory footprint on the web-scale dataset. However, these methods can only be applied in the look-up table manner for categorical features or numerical features after discretization. Few studies pay attention to the embedding learning of numerical features, which is vital in deep CTR models of the industry.

5.2 Feature Interaction

To capture the feature interactions effectively, different architectures are proposed in the research community of CTR prediction. According to different combinations of explicit and implicit feature
interaction, existing CTR models can be classified into two categories: *stacked structure* and *parallel structure*. Models with stacked structure first perform explicit feature interaction modeling over the embeddings and then stack a DNN to extract high-level implicit feature interactions. Representative models include FNN [30], IPNN [23], PIN [24], FiBiNET [10], FGCN [17], AutoGroup [18], DIN [33] and DIEN [32]. On the contrary, models with parallel structure leverage two parallel networks to capture explicit and implicit feature interactive signals respectively, and fuse the information in the output layer. Classic members in this category are Wide & Deep [2], DeepFM [7], AutoFIS [19], DCN [28], xDeepFM [16] and AutoInt [26]. In these models, implicit feature interactions are extracted via a DNN model; while for explicit feature interaction modeling, Wide & Deep [2] uses handcrafted cross features, DeepFM and AutoFIS adopt a FM [25] structure, DCN uses a cross network, xDeepFM employs a Compressed Interaction Network (CIN) and AutoInt leverages a multi-head self-attention network.

### 6 CONCLUSION

In this paper, we propose AutoDis, a pluggable embedding learning framework for numerical features in CTR prediction. AutoDis overcomes the shortcomings of the state-of-the-art processing methods and achieves better performance. It gains the improvement from these advantages: (1) High model capacity: The meta-embeddings shared by intra-field features capture global informative knowledge; (2) End-to-end training: The differentiable automatic discretization performs soft discretization, so that AutoDis can be optimized jointly with the ultimate goal of deep CTR models; (3) Unique representation: The weighted-average aggregation ensures that each feature learns a unique *Continuous-But-Different* representation. Extensive experiments on three real-world datasets and online A/B test demonstrate the effectiveness and compatibility of AutoDis.

### ACKNOWLEDGEMENT

Weinan Zhang is supported by “New Generation of AI 2030” Major Project (2018AA0100900) and National Natural Science Foundation of China (62076161, 61632017). This work is also sponsored by Extensive experiments on three real-world datasets and online A/B feature interactive signals respectively, and fuse the information in the output layer. Classic members in this category are Wide & Deep [2], DeepFM [7], AutoFIS [19], DCN [28], xDeepFM [16] and AutoInt [26]. In these models, implicit feature interactions are extracted via a DNN model; while for explicit feature interaction modeling, Wide & Deep [2] uses handcrafted cross features, DeepFM and AutoFIS adopt a FM [25] structure, DCN uses a cross network, xDeepFM employs a Compressed Interaction Network (CIN) and AutoInt leverages a multi-head self-attention network.

### 6 CONCLUSION

In this paper, we propose AutoDis, a pluggable embedding learning framework for numerical features in CTR prediction. AutoDis overcomes the shortcomings of the state-of-the-art processing methods and achieves better performance. It gains the improvement from these advantages: (1) High model capacity: The meta-embeddings shared by intra-field features capture global informative knowledge; (2) End-to-end training: The differentiable automatic discretization performs soft discretization, so that AutoDis can be optimized jointly with the ultimate goal of deep CTR models; (3) Unique representation: The weighted-average aggregation ensures that each feature learns a unique *Continuous-But-Different* representation. Extensive experiments on three real-world datasets and online A/B test demonstrate the effectiveness and compatibility of AutoDis.

### ACKNOWLEDGEMENT

Weinan Zhang is supported by “New Generation of AI 2030” Major Project (2018AA0100900) and National Natural Science Foundation of China (62076161, 61632017). This work is also sponsored by Extensive experiments on three real-world datasets and online A/B test demonstrate the effectiveness and compatibility of AutoDis.

### REFERENCES

[1] 2020. MindSpore. https://www.mindspore.cn/

[2] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hirshy Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Isipr, Rohan Aml, Zakaria Hause, Lichan Hong, Viyan Jain, Xiaolong Liu, and Yehuda Shah. 2016. Wide & Deep Learning for Recommender Systems. In Proc. Workshop Deep Learning for Recommender Systems.

[3] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In RecSys.

[4] Antonio Ginart, Maxim Naumov, Dheevatsa Mudigere, Jiyan Yang, and James Zou. 2019. Mixed dimension embeddings with application to memory-efficient recommendation systems. arXiv preprint arXiv:1909.11810 (2019).

[5] Krzysztof Grabarewicz and Norbert Jankowski. 2005. Feature Selection with Decision Tree Criterion. In IEEE Computer Society. 212–217.

[6] Yulong Gu, Zhaoyue Ding, Shuqiang Wang, Lixin Zou, Yiding Liu, and Dawei Yin. 2020. Deep MultiFaced Transformers for Multi-objective Ranking in Large-Scale E-commerce Recommender Systems. In CIKM. ACM. 2499–2500.

[7] Huifeng Guo, Ruiming Tang, Yuning Tang, Yeqing Li, ZhenguoGuo, and XiuqiangHe. 2017. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. In IJCAI.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In ICCV. 1026–1034.

[9] Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, Bo Liu, Tao Xu, Yanxin Shi, Antoine Atallah, Ralf Herbrich, Stuart Bowers, and Joaquin Quiñonero Candela. 2014. Practical Lessons from Predicting Clicks on Ads at Facebook. In IJCAI. ACM.

[10] Tongwen Huang, Zhiqi Zhang, and Junlin Zhang. 2019. FiBiNET: combining feature importance and bilinear feature interaction for click-through rate prediction. In RecSys. ACM. 169–177.

[11] Manas R. Joglekar, Jay K. Adams, Pranav Khatian, and Quoc V. Le. 2019. Neural Input Search for Large Scale Recommendation Models. CoRR abs/1907.04471 (2019). arXiv:1907.04471. http://arxiv.org/abs/1907.04471.

[12] Yu-Chin Jwoan, Yuan Zhonga, Wei-Shing Ching, and Chih-Jen Lin. 2016. Field-aware Factorization Machines for CTR Prediction. In RecSys.

[13] Bing-Cheng Kang, Derek Zhiyuan Cheng, Tianzheng Yao, Xinyang-Yi, Ting Chen, Lichan Hong, and Ed H. Chi. 2020. Deep Hash Embedding for Large-Vocab Categorical Feature Representations. CoRR abs/2010.10784 (2020).

[14] Guolin Ke, Zhenhui Xu, Jia Zhang, Jian Biao, and Tie-Yan Liu. 2019. DeepGBM: A Deep Learning Framework Distilled by GBDT for Online Prediction Tasks. In SIGKDD. ACM. 384–394.

[15] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014).

[16] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. xDeepFM: Combining Explicit and Implicit Feature Interactions for Recommender Systems. In SIGKDD.

[17] Bin Liu, Ruiming Tang, Yinghui Chen, Jinjua Yang, Huifeng Guo, and Yuan Zhou. 2019. Feature Generation by Convolutional Neural Network for Click-Through Rate Prediction. In WWW.

[18] Bin Liu, Niannan Xue, Huifeng Guo, Ruiming Tang, Stefanos Zafeiriou, Xiuqiang He, and Zhenguo Li. 2020. AutoGroup: Automatic Feature Grouping for Modelling Explicit High-Order Feature Interactions in CTR Prediction. In SIGIR. ACM.

[19] Bin Liu, Chenxu Zhu, Guolin Li, Weinan Zhang, Jincan Li, Ruiming Tang, Xi-uqiang He, and Yong Yu. 2020. AutoFIS: Automatic Feature Interaction Selection in Factorization Models for Click-Through Rate Prediction. In SIGKDD.

[20] H Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, et al. 2013. Ad click prediction: a view from the trenches. In SIGKDD. ACM. 1222–1230.

[21] Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, and Xiang Conde. 2019. Mixed dimension embeddings with application to memory-efficient recommendation systems. In Proc. Workshop Deep Learning for Recommender Systems.

[22] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In RecSys.

[23] Antonio Ginart, Maxim Naumov, Dheevatsa Mudigere, Jiyan Yang, and James Zou. 2019. Mixed dimension embeddings with application to memory-efficient recommendation systems. arXiv preprint arXiv:1909.11810 (2019).

[24] Krzysztof Grabarewicz and Norbert Jankowski. 2005. Feature Selection with Decision Tree Criterion. In IEEE Computer Society. 212–217.

[25] Yulong Gu, Zhaoyue Ding, Shuqiang Wang, Lixin Zou, Yiding Liu, and Dawei Yin. 2020. Deep MultiFaced Transformers for Multi-objective Ranking in Large-Scale E-commerce Recommender Systems. In CIKM. ACM. 2499–2500.

[26] Huifeng Guo, Ruiming Tang, Yuning Tang, Yeqing Li, ZhenguoGuo, and XiuqiangHe. 2017. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. In IJCAI.