Numerical Feature Representation with Hybrid N-ary Encoding

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

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

Weiwen Liu
liuweiwen8@huawei.com
Huawei Noah's Ark Lab

Yue Ding
dingyue@sjtu.edu.cn
Shanghai JiaoTong University

Yunzhe Li
liyzh28@sjtu.edu.cn
Shanghai JiaoTong University

Wei Guo
guowei67@huawei.com
Huawei Noah's Ark Lab

Yichao Wang
wangyichao5@huawei.com
Huawei Noah's Ark Lab

Zhicheng He
hezhicheng9@huawei.com
Huawei Noah's Ark Lab

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

Rui Zhang
rayteam@yeah.net
ruizhang.info

ABSTRACT

Numerical features (*e.g.*, statistical features) are widely used in recommender systems and online advertising. Existing approaches for numerical feature representation in industry are primarily based on discretization. However, hard-discretization based methods (*e.g.*, Equal Distance Discretization) are deficient in continuity while soft-discretization based methods (*e.g.*, AutoDis) lack discriminability. To emphasize both *continuity* and *discriminability* for numerical features, we propose an end-to-end representation learning framework named **NaryDis**. Specifically, NaryDis first leverages hybrid n-ary encoding as an automatic discretization module to generate hybrid-grained discretization results (multiple encoded sequences). Each position of the encoded sequence is assigned with a positional embedding and an intra-ary attention network is leveraged to aggregate the positional embeddings for obtaining ary-wise representations. Then an inter-ary attention is adopted to assemble these representations, which are further constrained by a self-supervised regularization module. Comprehensive experiments on two public datasets are conducted to show the superiority and compatibility of NaryDis. Besides, we deeply investigate the properties of continuity and discriminability. Moreover, we further verify the effectiveness of NaryDis on a large-scale industrial advertisement dataset.

CCS CONCEPTS

• **Information systems** → **Information retrieval**; **Recommender systems**; • **Computing methodologies** → **Machine learning**.

KEYWORDS

Numerical Features, Representation Learning, Click-Through Rate Prediction, Neural Network

ACM Reference Format:
Bo Chen, Huifeng Guo, Weiwen Liu, Yue Ding, Yunzhe Li, Wei Guo, Yichao Wang, Zhicheng He, Ruiming Tang, and Rui Zhang. 2022. Numerical Feature Representation with Hybrid N-ary Encoding. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM '22), October 17–21, 2022, Atlanta, GA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3511808.3557090

1 INTRODUCTION

Click-Through Rate (CTR) prediction [29] is critical for industrial recommender systems [8, 14] and online advertising [23, 26], which estimates the probability of a user clicking an item in a certain context. Existing deep CTR models follow the *Embedding & Feature Interaction* (FI) paradigm [2, 11, 37], where the Embedding module, as the cornerstone for CTR models [17], is responsible for producing informative feature representations and the FI module specializes in effectively modeling feature interactions [41]. The features used in industrial CTR prediction scenarios include *categorical features* and *numerical features*. The domain of value is enumerable for categorical features (*e.g.*, `Gender` = {*male*, *female*}) while infinite for numerical features (*e.g.*, `Number of Clicks` = $\{0, 1, 2, \dots\}$). Besides, there is a magnitude relationship between the feature values (*e.g.*, 10 is larger than 2) for numerical features, which is untenable for categorical features. Due to the infinity, continuity as well as magnitude property of numerical features, it is essential for the representation of the numerical features to retain ***continuity*** for similar features as well as ***discriminability*** for distinct features.

Existing research for numerical feature representation can be categorized into two groups: *non-discretization* and *discretization*. The former uses the transformation of features as representations, which suffer from low capacity and compatibility. The latter is commonly used in industrial scenarios [18, 25], which can be further divided into two sub-categories: 1) ***Hard-discretization*** that allocates each feature into *a single deterministic bucket*; 2) ***Soft-discretization*** that maps each feature into *a probabilistic distribution over multiple*

*Co-first authors with equal contributions.
†Corresponding author.

CIKM '22, October 17–21, 2022, Atlanta, GA, USA

ACM ISBN 978-1-4503-9236-5/22/10…$15.00
https://doi.org/10.1145/3511808.3557090

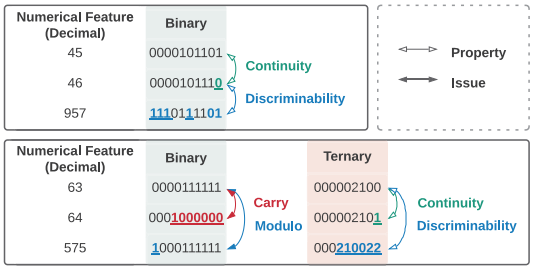


Figure 1: The subgraph above illustrates the continuity and decriminability of the numerical system; the subgraph below illustrates the "*collision risks*" and the solution.

buckets. Hard-discretization methods leverage pre-defined rules or pre-trained models to divide features into a deterministic bucket, which cannot guarantee either continuity or discriminability. Recently proposed AutoDis [10] uses an automatic soft-discretization module, projecting each feature into a probabilistic distribution over multiple buckets, ensuring superior continuity. However, this approach has poor discriminability (evidence provided in Section 4.3). The reason for this situation is that the automatic discretization module parameterized by a neural network is an implicit black-box procedure and the projecting distribution is highly dominated by the weights of neurons.

In fact, the numbers themselves give us a clue to emphasize both continuity and discriminability, *i.e.*, the **positional numeral system** [5]. In the binary numeral system, all integers can be represented as a unique sequence of digits (non-collision), where each position takes one digit from a standard digit set $\mathcal{D}_2 = \{0, 1\}$. For example, integers in $[0, 1024)$ can be represented as 10 digits such as $\underline{0}_{2^9} \ldots \underline{0}_{2^6} \underline{1}_{2^5} \underline{0}_{2^4} \underline{1}_{2^3} \underline{1}_{2^2} \underline{0}_{2^1} \underline{1}_{2^0}$ for feature value "45", which can be regarded as a discretization. Moreover, the encoding process is deterministic and non-parametric, which can be computed on-the-fly. As shown in the upper subgraph in Figure 1, discretization via positional numeral system not only ensures the superior continuity of discretized results *w.r.t.* the original numerical features (*e.g.*, similar feature values "45" and "46" have similar sequences of digits), but markedly improves the discriminability (*e.g.*, the sequences of digits for distinct feature values "46" and "957" are distinct). Nevertheless, using the single n-ary positional numeral system for discretization is confronted with two individual "*collision risks*" (shown as the lower subgraph in Figure 1):

- *Carry*. Numbers around the carry border (power of base) are usually converted to dis-similar discretized results despite their similar values, such as values "63" and "64".
- *Modulo congruence*. Numbers that modulo congruent to a high positional value are usually converted to similar discretized results despite their dis-similar values, such as $63 \equiv 575 \ (mod\ 2^9)$.

Fortunately, both risks can be easily solved by using hybrid n-ary positional numeral systems as they have diverse carry borders and positional values. For example, the carry borders and positional values are $\{2^0, 2^1, 2^2, \cdots\}$ in binary, while $\{3^0, 3^1, 3^2, \cdots\}$ in ternary. Therefore, the "*collision risks*" existing in binary may be solved in ternary, which is shown in the lower subgraph in Figure 1.

Inspired by the hybrid n-ary positional numeral systems, we propose an end-to-end automatic discretization and representation learning framework for numerical features (NaryDis), which emphasizes both *continuity* and *discriminability*. The architecture of NaryDis is depicted in Figure 3. Specifically, NaryDis utilizes hybrid n-ary encoding to generate multiple sequences for each feature in a hybrid-grained manner, so that different encoded sequences will complement each other to alleviate the "*collision risks*" mentioned above. Then each position of the encoded sequence is assigned with a shared positional embedding for modeling global knowledge. To identify the importance of different positions in the sequence, we leverage a light-weight intra-ary attention network to distinguish and aggregate those positional embeddings, thus obtaining the corresponding ary-wise representation. Finally, an inter-ary attention network is adopted to assemble these ary-wise representations for learning the final representation. Furthermore, an auxiliary self-supervised regularization module is introduced to provide constraint signals because different ary-wise representations of the same numerical feature can be regarded as various views.

Our main contributions are summarized as follows:

- We propose NaryDis, a hybrid n-ary encoding-based automatic discretization and representation learning framework for numerical features, which ensures prominent *continuity* and *discriminability* in an end-to-end manner.
- In NaryDis, we utilize the hybrid n-ary encoding to discretize the numerical features with different granularities. Besides, hierarchical intra-ary and inter-ary attention are elaborated to discriminate the importance of different positions and n-ary encodings, respectively. Moreover, an auxiliary self-supervised regularization module is designed to provide constraint signals.
- Comprehensive experiments on two publicly available datasets are conducted to demonstrate the superiority of the NaryDis framework. In addition, NaryDis is compatible with various deep CTR models by improving their performance significantly, and we deeply investigate the properties of continuity and discriminability. Moreover, we further verify the effectiveness over a large-scale industrial advertisement dataset.

2 RELATED WORK

Numerical features are widely used in recommender systems and online advertising. Existing representation learning approaches for numerical features can be categorized into two groups: *non-discretization* and *discretization*. The *non-discretization* methods do not discretize the features but transform them via various functions, which suffer from low capacity and compatibility [10]. DMT [9] and YouTube DNN [6] use the normalized, square, and square root of numerical features as the representations directly without learning embeddings. Differently, DLRM [24] adopts a neural network to transform all numerical features into an instance-level embedding while DeepFM [11] and AutoInt [30] share a uniform field-level embedding (FE) for all the intra-field features and then multiply the FE with their feature values.

Discretization-based approaches, are extensively used in industrial scenarios [18, 25], which can be further divided into two

sub-categories: ***hard-discretization*** and ***soft-discretization***. Concretely, hard-discretization allocates each feature into a single deterministic bucket via pre-defined rules or pre-trained models, such as Equal Distance/Frequency Discretization [25], Logarithm-based Discretization [18], and Tree-based Discretization [7, 19]. These hard-discretization methods cannot be optimized with the ultimate prediction goal, as well as lack continuity and discriminability. Additionally, soft-discretization is proposed by AutoDis [10], which maps each feature into a probabilistic distribution via a differentiable discretization module – a neural network, ensuring superior continuity. However, the projecting procedure via a neural network is an implicit black-box and the allocation probability is highly dominated by the weights of neurons, resulting in unsatisfying discriminability. To guarantee both continuity and discriminability, we propose a hybrid n-ary encoding-based automatic discretization and representation learning framework NaryDis.

3 METHODOLOGY

3.1 Vanilla Model

3.1.1 Input. Specifically, the dataset for training CTR models consists of instances $(\mathbf{x}, y)$, where each input record includes P numerical fields[1] and Q categorical fields $\mathbf{x} = [\underbrace{x_1, x_2, \ldots, x_P}_{\text{scalars}}; \underbrace{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_Q}_{\text{one-hot vectors}}]$, and $y \in \{1, 0\}$ is the label.

3.1.2 Embedding. For the i-th categorical field, the feature embedding can be obtained by embedding look-up operation [39]: $\mathbf{e}_i = \mathbf{x}_i \cdot \mathbf{E}_i$, where $\mathbf{E}_i \in \mathbb{R}^{n_i \times k}$ is the embedding table, n_i and k are the vocabulary size and embedding size, respectively. For the i-th numerical field, the embedding can be obtained by various approaches: $\mathbf{e}_i = \texttt{Func\_num}(x_i)$, where $\texttt{Func\_num}(\cdot)$ indicates a specific numerical feature representation method, such as Equal Distance Discretization [25], AutoDis [10] and NaryDis.

Inspired by the hybrid n-ary positional numeral systems, we innovatively propose a binary encoding-based automatic discretization module, which is shown in Figure 2. Specifically, a numerical feature x_i is first encoded into a sequence of d-dimensional binary digits $\mathbf{X}_i \in \{0, 1\}^d$ via the decimal-binary conversion procedure [28]. Then the soft discretization distribution $\widehat{\mathbf{X}}_i \in \mathbb{R}^d$ is obtained by the normalization of L_1-norm, *i.e.*, $\widehat{\mathbf{X}}_i = \frac{\mathbf{X}_i}{\|\mathbf{X}_i\|_1}$, where the j-th element $\widehat{x}_{ij}$ represents the selection probability of the j-th position.

To capture the shared global knowledge, each ***position*** (*a.k.a.*, ***bucket***) in the binary sequence is assigned with a ***positional embedding*** (*a.k.a.*, ***bucket embedding***). These positional embeddings form a field-specific positional embedding matrix $\mathbf{E}_i \in \mathbb{R}^{d \times k}$. Finally, the representation of feature x_i can be obtained by the aggregation operation $f_{\text{Agg}}(\cdot)$:

$$\mathbf{e}_i = \texttt{Func\_num}(x_i) = f_{\text{Agg}}(\widehat{\mathbf{X}}_i, \mathbf{E}_i). \tag{1}$$

To simplify, we use weighted pooling as the aggregation operation.

Similar to AutoDis, binary encoding-based automatic discretization also performs soft-discretization with discretized distribution $\widehat{\mathbf{X}}_i$. The decimal-binary conversion process not only ensures the superior continuity of discretization results *w.r.t.* the numerical features, but markedly improves the discriminability.

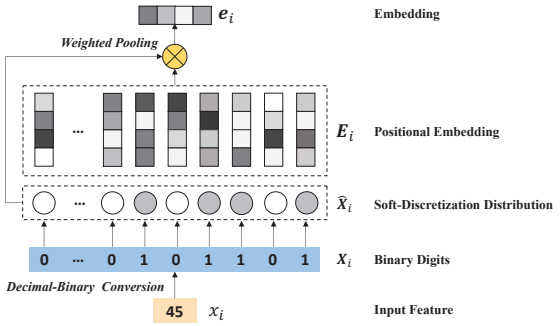


Figure 2: The binary encoding-based discretization module.

3.1.3 FI & Prediction. Based on the generated embeddings, the CTR prediction is obtained through different FI module [32, 40]:

$$\hat{y} = \texttt{Func\_FI}([\underbrace{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_P}_{\text{numerical embed}}; \underbrace{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_Q}_{\text{categorical embed}}]), \tag{2}$$

where $\hat{y}$ is the predicted CTR, $\texttt{Func\_FI}(\cdot)$ indicates the FI module of different CTR models, such as inner product in IPNN [25], Multilayer Perceptron (MLP) in DeepFM [11], feature crossing in DCN [34], and automated feature interaction functions [41].

3.2 NaryDis

In this section, we elaborately present NaryDis, a pluggable representation framework for numerical features that is compatible with various deep CTR models. The architecture of NaryDis is depicted in Figure 3, which contains four core modules. Specifically, a *hybrid n-ary encoding* module is proposed to generate multiple sequences for each feature in a hybrid-grained manner. Then hierarchical *intra-ary* and *inter-ary attention* modules are utilized to distinguish the importance of different positions in each sequence and n-ary encodings, respectively. Moreover, a *self-supervised regularization* module is deployed to constrain the ary-wise representations and jointly optimize with the primary task.

3.2.1 Hybrid n-ary Encoding. Although binary encoding-based discretization contributes to achieving both continuity and discriminability, there exist two kinds of potential "*collision risks*". As shown in Figure 1, influenced by the *carry*, though values "63" and "64" are quite close, their discretized results under binary encoding are significantly different (7 digits are opposed). Whereas values "63" and "575" share similar results (only 1 bit is flipped) though their values are extremely different due to the *modulo congruence*. To overcome these risks existing in the single n-ary encoding situation, we introduce hybrid n-ary encoding. Specifically, even though some similar values have different results after binary encoding, their results under other encodings (*e.g.*, ternary) may be similar. Meanwhile, some distinct values may share a large proportion of identical results in binary encoding, but their results under other encoding settings may be completely different, which is illustrated in Figure 1.

[1] We use "field" to represent a class of features following [18] and "feature" to represent a value of a specific field, *e.g.*, `Gender` is a field and female is a feature of `Gender` field.

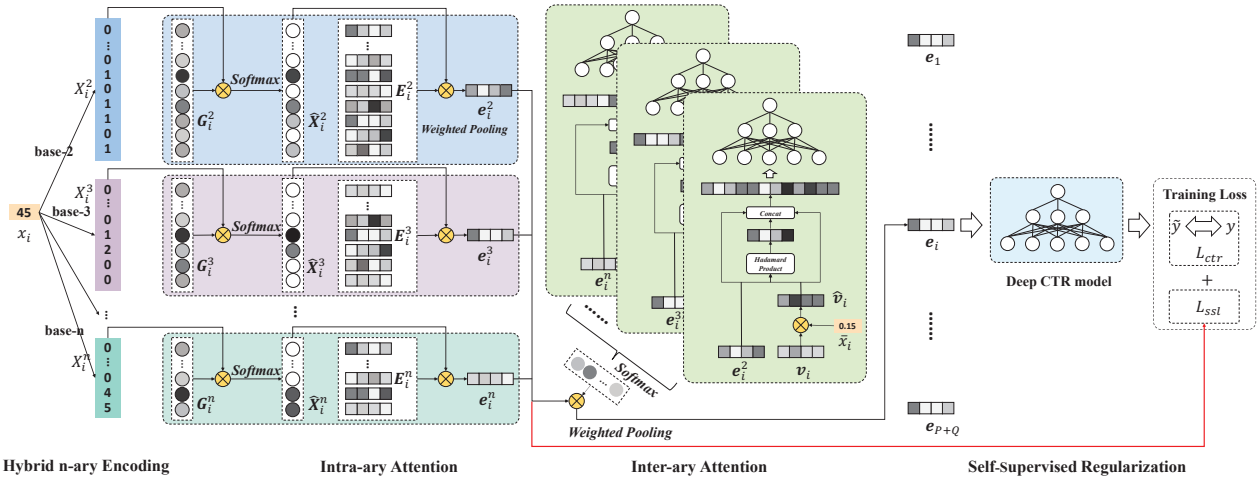


Figure 3: The architecture of NaryDis.

For the purpose of eliminating the "*collision risks*" in single n-ary encoding-based discretization, hybrid n-ary encoding is utilized. Concretely, NaryDis leverages different encoding setting $\mathcal{N}$ to convert feature x_i into multiple sequences: namely $\mathcal{X}_i = \{\mathbf{X}_i^2, \mathbf{X}_i^3, \dots\}$, where $\mathbf{X}_i^n$ denotes the n-ary encoded sequence (*a.k.a.*, base-n, where n is an integer greater than 1). The length of the sequence $\mathbf{X}_i^n$ is represented as d_n, namely $\mathbf{X}_i^n = [x_{i1}^n, x_{i2}^n, \cdots, x_{id_n}^n] \in \{0, 1, \dots, n-1\}^{d_n}$.

3.2.2 Intra-ary Attention. For a n-ary encoded sequence $\mathbf{X}_i^n$, each *position* can be regarded as a *discretized bucket* in the hard-discretization methods. Inspired by the gating mechanism used in MMoE [22], NaryDis proposes a gate-based intra-ary attention network to discriminate the importance of different positions. In particular, a **light-weight** field-wise gating unit $\mathbf{G}_i^n = [\mathbf{g}_{i1}^n, \mathbf{g}_{i2}^n, \cdots, \mathbf{g}_{id_n}^n]$ is proposed, where $\mathbf{g}_{ij}^n$ is a scalar that represents the gating weight for the j-th position of the n-ary encoded sequence. Then the gating unit $\mathbf{G}_i^n$ is multiplied with sequence $\mathbf{X}_i^n$ before being normalized via a $\texttt{Softmax}$ function. Thus, we attain the soft discretization distribution under the n-ary encoding, which is jointly determined by the encoded sequence and the gate-based attention network: $\widehat{\mathbf{X}}_i^n = \texttt{Softmax}(\texttt{Mask}(\mathbf{X}_i^n \odot \mathbf{G}_i^n))$, where $\odot$ is the Hadamard product. Each element $\widehat{x}_{ij}^n$ is defined as:

$$\widehat{x}_{ij}^n = \texttt{Softmax}(x_{ij}^n \mathbf{g}_{ij}^n) = \frac{\exp(x_{ij}^n \mathbf{g}_{ij}^n / \tau)}{\sum_{j'=1}^{d_n} \exp(x_{ij'}^n \mathbf{g}_{ij'}^n / \tau)}, \tag{3}$$

where τ is the temperature coefficient to control the distribution. Besides, to learn the globally shared intra-field knowledge, a field-specific positional embedding matrix $\mathbf{E}_i^n \in \mathbb{R}^{d_n \times k}$ is assigned for the i-th field with the n-ary encoding. Hence, the representation of feature x_i under the n-ary encoding can be expressed as:

$$\mathbf{e}_i^n = \sum_{j=1}^{d_n} \widehat{x}_{ij}^n \mathbf{E}_{ij}^n. \tag{4}$$

3.2.3 Inter-ary Attention. After obtaining the ary-wise representations under different encoding setting $\mathcal{N}$, *i.e.*, $\mathcal{E}_i = \{\mathbf{e}_i^2, \mathbf{e}_i^3, \dots\}$, the next step is to assemble them and obtain the final representation. One of the most pervasive approaches is to select an optimal representation via AutoML-based methods like Gumbel-Softmax [16]. However, we argue that this hard-selection approach is sub-optimal because selecting a single representation may encounter the above-mentioned "*collision risks*". Additionally, hybrid n-ary encoding can be regarded as discretization with different granularities and these representations are conducive to depicting features comprehensively.

Therefore, NaryDis leverages an inter-ary attention network to soft-select and assemble these representations adaptively *w.r.t.* the feature field. It is noteworthy that the soft-selection process is field-aware because the importance of the same feature value is different for various fields, such as feature "25" in `Temperature` field and `Price` field. Concretely, in order to take both feature field and feature value into consideration, we multiply the normalized feature value $\bar{x}_i$ with a shared field-wise vector $\mathbf{v}_i \in \mathbb{R}^k$ to obtain a feature-aware vector $\widehat{\mathbf{v}}_i \in \mathbb{R}^k$. Then, an attention network is leveraged to discriminate the contributions of different representations by calculating their relevance score w_i^n:

$$w_i^n = \texttt{MLP}([\mathbf{e}_i^n || \widehat{\mathbf{v}}_i || (\mathbf{e}_i^n \odot \widehat{\mathbf{v}}_i)]), \tag{5}$$

where $||$ denotes the concatenation operation and $\texttt{MLP}(\cdot)$ is a three-layer neural network with $\texttt{ReLU}$ activation function. Finally, the representation of feature x_i can be obtained by an attentive pooling whose weights are calculated by normalizing the above relevance scores using $\texttt{Softmax}$, namely:

$$\mathbf{e}_i = \sum_{n \in \mathcal{N}} \texttt{Softmax}(w_i^n) \mathbf{e}_i^n = \sum_{n \in \mathcal{N}} \frac{\exp(w_i^n)}{\sum_{n' \in \mathcal{N}} \exp(w_i^{n'})} \mathbf{e}_i^n, \tag{6}$$

where $\mathcal{N}$ is the selected encoding setting.

Finally, the embeddings of both categorical and numerical features are concatenated and fed into a deep CTR model to obtain the predicted CTR $\hat{y}$, as shown in Equation (2).

3.2.4 Self-supervised Regularization. The ary-wise representations of feature x_i under different encoding setting $\mathcal{E}_i = \{\mathbf{e}_i^2, \mathbf{e}_i^3, \dots\}$ can be regarded as $|\mathcal{N}|$ different views, which are different data augmentation for the same feature natively and supposed to be close in the latent space. Therefore, we adopt contrastive learning to provide self-supervised regularization signals, constraining different representations in $\mathcal{E}_i$ to tend to be similar. Specifically, we

first sample a pair of representations from $\mathcal{E}_i$, which are two data augmentation views for the same feature x_i, namely:

$$\mathbf{e}_i^p \leftarrow \texttt{sample}(\mathcal{E}_i), \mathbf{e}_i^q \leftarrow \texttt{sample}(\mathcal{E}_i). \tag{7}$$

Then a three-layer neural network with ReLU activation function is leveraged to transform the representations into a hidden space and obtain a pair of hidden vectors, that is:

$$\mathbf{z}_i^p = \texttt{MLP}(\mathbf{e}_i^p), \mathbf{z}_i^q = \texttt{MLP}(\mathbf{e}_i^q). \tag{8}$$

Hence, we treat $(\mathbf{z}_i^p, \mathbf{z}_i^q)$ as positive pairs and $(\mathbf{z}_i^p, \mathbf{z}_{i'}^q)$ as negative pairs for $i \neq i'$.

We follow the SimCLR framework [3] that is widely applied in computer vision [13] and recommendation [36, 38] and use the contrastive loss InfoNCE [12] to maximize the similarity of positive pairs and minimize that of negative pairs by:

$$\mathcal{L}_{ssl} = -\frac{1}{Q}\sum_{i=1}^{Q} \log \frac{\exp(s(\mathbf{z}_i^p, \mathbf{z}_i^q)/\tau_{ssl})}{\sum_{i'=1}^{Q} \exp(s(\mathbf{z}_i^p, \mathbf{z}_{i'}^q)/\tau_{ssl})}, \tag{9}$$

where τ_{ssl} is the softmax temperature and $s(\cdot)$ measures the similarity between two hidden vectors, which is calculated by $s(\mathbf{z}_i^p, \mathbf{z}_i^q) = \langle \mathbf{z}_i^p, \mathbf{z}_i^q \rangle / (\|\mathbf{z}_i^p\| \cdot \|\mathbf{z}_i^q\|)$. Q is the total number of training instances.

3.3 Training

To optimize the CTR prediction task [29] as well as enable the similarity regularization of contrastive learning, we adopt a multi-task training paradigm where the main supervised CTR task and the auxiliary regularization task are jointly optimized:

$$\mathcal{L} = \mathcal{L}_{ctr} + \alpha \cdot \mathcal{L}_{ssl}, \tag{10}$$

where α controls the self-supervised regularization strength. The loss function of CTR main task is the widely-used LogLoss with a regularization term:

$$\mathcal{L}_{ctr} = -\frac{1}{Q}\sum_{i=1}^{Q} y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i) + \lambda \|\Theta\|_2, \tag{11}$$

where y_i and $\hat{y}_i$ are the ground truth label and the predicted CTR of the i-th instance, respectively. Hyper-parameter λ is the L_2 regularization weight. $\Theta = \{\Theta_{Cat\_Emb}, \Theta_{NaryDis}, \Theta_{CTR}\}$ are the parameters of feature embeddings in categorical fields, parameters of NaryDis (*i.e.*, Eq.(3-9)) as well as deep CTR model (*i.e.*, Eq.(2)).

3.4 Discussion

3.4.1 Connections and differences with existing methods. In this subsection, we discuss the connections and differences between NaryDis and the existing methods. For the *Non-Discretization* methods, YouTube [6] and DLRM [24] transform the numerical features directly without learning embeddings. Besides, the field-level embedding (FE) [11, 30] can be viewed as a special case of NaryDis, which corresponds to the base of infinity (*i.e.*, base-∞). In this case, the encoded sequence X_i^∞ only contains one position. For the *Hard-Discretization* methods [18, 19, 25], each feature is discretized into a single deterministic bucket via pre-defined rules or pre-trained models. Instead, NaryDis is a soft discretization solution, which discretizes each feature into multiple buckets (positions) with a probabilistic distribution. For the *Soft-Discretization* method AutoDis [10], the discretized distribution is achieved by a neural network, which cannot guarantee discriminability. NaryDis, on the contrary, leverages the hybrid n-ary encoding to perform mixed-granularity discretization, ensuring prominent continuity and discriminability.

Table 1: Statistics of evaluation datasets.

Dataset	#Num Fields	#Cats Fields	#Instances ($\times 10^6$)	Positive Ratio
Criteo	13	26	45.8	25.6%
AutoML	23	51	4.69	5.8%

3.4.2 Complexity Analysis. In this subsection, we analyze the complexity of NaryDis. Suppose the embedding size is denoted as k, we select M different encoding setting (*i.e.*, $|\mathcal{N}| = M$) and set d as the maximum length of encoded sequences. Hence, for a numerical feature, the complexity of NaryDis is $O(Mdk)$. Our experimental study in Section 4.4 shows that good performance can be achieved at a small $M(3 \sim 4)$. Besides, the length of encoded sequences d is also small ($10 \sim 20$) since a small exponent can represent a large range of values, *e.g.*, $5^{10} = 9765625$. Therefore, the complexity of the NaryDis is comparable to AutoDis, whose complexity is $O(hk)$, where h denotes the number of meta-embeddings (buckets).

3.4.3 Compatibility Analysis. NaryDis is a model-agnostic framework for numerical feature representation learning, which can be applied seamlessly to mainstream embedding-based deep models, such as DNN [39], DCN [34], PNN [25] and *etc.* Furthermore, some models from the Factorization Machines family [27] (*e.g.*, FFM [18] and DeepFM [11]) that requires equal-length embeddings are also applicable. This framework can be generalized well to various deep models and the compatibility study is elaborated in Section 4.2.2.

4 EXPERIMENTS

4.1 Experimental Setting

4.1.1 Datasets. To evaluate the effectiveness of the proposed NaryDis, we conduct extensive experiments on two popular benchmarks: *Criteo*, *AutoML*, whose statistics are summarized in Table 1.

- Criteo dataset (with 13 numerical feature fields, where the feature values are non-negative integers) is published in Criteo Display Advertising Challenge 2013[2], which is extensively used in evaluating CTR models.
- AutoML dataset is published from "AutoML for Lifelong Machine Learning" Challenge in NeurIPS 2018[3], containing 23 numerical feature fields where the feature values include (positive and negative) integers and floating-point numbers.

4.1.2 Evaluation Protocols. We leverage **AUC** and **LogLoss**, which are the commonly used metrics in CTR prediction, to evaluate the performance. All the experiments are repeated 5 times by changing the random seeds. The two-tailed unpaired t-test [1] is performed to detect the significance of NaryDis.

4.1.3 Baselines. To demonstrate the effectiveness of NaryDis, we compare it with two categories of representation learning methods for numerical features:

(1) *Non-Discretization* methods

[2]https://www.kaggle.com/c/criteo-display-ad-challenge
[3]https://www.4paradigm.com/competition/nips2018

- **YouTube** that uses square and square root of the normalized numerical features [6], *i.e.*, $\mathbf{e} = [\tilde{x}_1^2, \tilde{x}_1, \sqrt{\tilde{x}_1}, \ldots, \tilde{x}_P^2, \tilde{x}_P, \sqrt{\tilde{x}_P}]$.
- **DLRM** that uses a multi-layer perception to transform all numerical features into an instance-level embeddings [24], *i.e.*, $\mathbf{e} = MLP([x_1, x_2, \ldots, x_P])$.
- **FE** that shares a uniform field-level embedding (FE) for all the intra-field features [11, 30], *i.e.*, $\mathbf{e} = [x_1 \cdot \mathbf{e}_1, \ldots, x_P \cdot \mathbf{e}_P]$.

(2) *Discretization* methods

Hard-Discretization:

- **EDD** (Equal Distance Discretization) partitions the feature values into equal-width buckets [25].
- **LD** (Logarithm-based Discretization) utilizes the logarithm and floor operation to transform the numerical features to categorical form [18], *i.e.*, $\widehat{x_i} = floor(log(x_i)^2)$.
- **TD** (Tree-based Discretization), *i.e.*, DeepGBM [19].

Soft-Discretization:

- **AutoDis** leverages a neural network to perform automatic soft discretization [10].
- **NaryDis** and the **Vanilla version** (in Section 3.1) are our proposed methods.

Moreover, to validate the compatibility of NaryDis framework with various embedding-based deep CTR models, we apply NaryDis to five representative models: **DNN** [39], **DeepFM** [11], **DCN** [34], **IPNN** [25], and **DCN-V2** [35].

4.1.4 Hyper-parameter Settings. According to the settings of AutoDis [10], we keep the same hyper-parameters for all the baselines, including embedding size and network architecture. Specifically, for NaryDis, all the numerical features are converted to integers. We magnify floating-point numbers to integers (*e.g.*, $0.35 \rightarrow 35$) and set up a sign bit for identifying positive and negative numbers. The default encoding setting is binary to decimal. All the models are optimized with mini-batch Adam [20], where the learning rate is searched from {1e-3, 2e-3, ..., 1e-2}. Batch Normalization [15] and Dropout [31] is applied to avoid overfitting. Besides, the hidden layers of MLP network is 80-40-1 in the inter-ary attention network and 64-64-64 in the self-supervised regularization module. The weight of L_2 regularization [21] λ is tuned in {1e-5, 5e-5, ..., 1e-2} and the self-supervised regularization strength α is tuned in $\{0.1, 0.2, \ldots, 1.2\}$. For the softmax temperature, we use an annealing scheme $\texttt{max}(0.1, 1 - 0.00005 \cdot t)$, where t is the training step.

4.2 Overall Performance

In this section, we evaluate the effectiveness and compatibility of NaryDis. We first compare NaryDis with the state-of-the-art representation learning methods and then apply NaryDis to five popular deep CTR models and report the improvements.

4.2.1 Comparison with other methods. We select DeepFM as the backbone CTR model and compare different numerical feature representation learning methods. The overall performance is reported in Table 2, from which we have the following observations:

- Firstly, comparing the *non-discretization* and *discretization* approaches, we find that *non-discretization* approaches perform generally worse than *discretization*. This suggests that suffering from low capacity, *non-discretization* methods are insufficient to capture the complex feature interactions, thereby limiting the performance improvement.
- Secondly, the performance of AutoDis verifies that *soft-discretization* can improve the quality of representation learning. As demonstrated in Table 2, AutoDis outperforms other methods by a large margin in terms of both AUC and LogLoss. The reason is that, rather than being decoupled from the model optimization like the *hard-discretization*, *soft-discretization* is jointly optimized with the deep CTR model. In addition, *soft-discretization* can better preserve the continuity *w.r.t.* the original value due to the probabilistic distribution. Nevertheless, the discretization results of AutoDis are less discriminative, even for very distinct feature values, thus leading to suboptimal representations.
- Thirdly, by simply incorporating the binary encoding, the vanilla version of NaryDis outperforms the state-of-the-art methods, which demonstrates the benefits of n-ary encoding. Additionally, the proposed NaryDis consistently yields the best performances on all datasets. For instance, NaryDis surpasses the strongest baseline AutoDis by 0.12% and 0.17% in terms of AUC on Criteo and AutoML datasets, respectively. Actually, a small increase at **0.001-level** in AUC is likely to yield a significant online CTR improvement and huge profits [4, 25]. We attribute this great improvement to the hybrid n-ary encoding that discretizes the features with different granularities, enhancing the expressiveness. Besides, the intra-ary and inter-ary attention are conducive to discriminating the importance of different positions and n-ary encodings, facilitating learning informative embeddings.

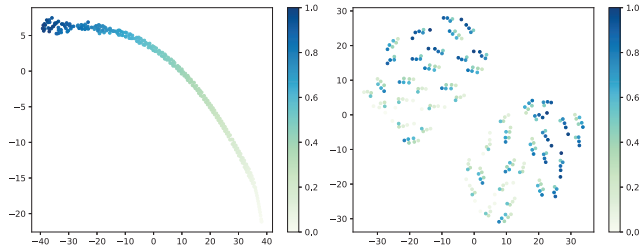
Table 2: The overall performance comparison. Boldface denotes the highest score and underline indicates the best result of the baselines. $\star$ represents significance level p-value < 0.05 of comparing NaryDis with the best baselines.

Category	Method	Criteo		AutoML	
		AUC	LogLoss	AUC	LogLoss
Non-Discretization	YouTube	0.8104	0.4437	0.7382	0.1928
	DLRM	0.8114	0.4404	0.7525	0.1900
	FE	0.8107	0.4412	0.7523	0.1899
Discretization	EDD	0.8125	0.4399	0.7545	0.1898
	LD	0.8138	0.4388	0.7527	0.1899
	TD	0.8130	0.4392	0.7531	0.1899
	AutoDis	<u>0.8152</u>	<u>0.4370</u>	<u>0.7562</u>	<u>0.1892</u>
NaryDis (vanilla)		0.8154	0.4368	0.7564	0.1892
NaryDis		**0.8162***	**0.4361***	**0.7575***	**0.1889***
Rel Improv.		**0.12%**	**0.21%**	**0.17%**	**0.16%**

4.2.2 Compatibility with different CTR models. Our proposed NaryDis is a flexible representation learning framework that is compatible with various deep CTR models. To verify the compatibility, we apply NaryDis and the strongest baseline AutoDis to a series of popular models to compare their performances on the public datasets Criteo and AutoML, whose results are presented in Table 3. We observe that compared with the field-level embedding (FE) [11], AutoDis, and NaryDis significantly improve the prediction performances of all models, which shows the effectiveness of the soft-discretization. Compared to AutoDis, NaryDis consistently outperforms. Moreover, the average AUC and Logloss gain are 0.23% and 0.27% respectively, verifying that NaryDis is compatible with various deep models and can steadily enhance their

Table 3: Compatibility Study of NaryDis.

Backbone Model	Method	Criteo		AutoML	
		AUC	LogLoss	AUC	LogLoss
DNN	FE	0.8059	0.4456	0.7383	0.1926
	AutoDis	0.8092	0.4426	0.7453	0.1907
	NaryDis	**0.8133***	**0.4390***	**0.7465***	**0.1906**
	Rel Improv.	**0.51%**	**0.81%**	**0.16%**	**0.05%**
DeepFM	FE	0.8107	0.4412	0.7523	0.1899
	AutoDis	0.8152	0.4370	0.7562	0.1892
	NaryDis	**0.8162***	**0.4361***	**0.7575***	**0.1889***
	Rel Improv.	**0.12%**	**0.21%**	**0.17%**	**0.16%**
DCN	FE	0.8091	0.4425	0.7489	0.1909
	AutoDis	0.8132	0.4395	0.7511	0.1898
	NaryDis	**0.8137***	**0.4384***	**0.7540***	**0.1894***
	Rel Improv.	**0.06%**	**0.25%**	**0.39%**	**0.21%**
IPNN	FE	0.8101	0.4415	0.7519	0.1896
	AutoDis	0.8136	0.4383	0.7541	0.1894
	NaryDis	**0.8157***	**0.4365***	**0.7552***	**0.1892***
	Rel Improv.	**0.26%**	**0.41%**	**0.15%**	**0.11%**
DCN-V2	FE	0.8093	0.4423	0.7510	0.1902
	AutoDis	0.8135	0.4385	0.7522	0.1898
	NaryDis	**0.8148***	**0.4373***	**0.7549***	**0.1893***
	Rel Improv.	**0.16%**	**0.27%**	**0.36%**	**0.26%**



(a) AutoDis (b) Binary

(c) Novenary (d) NaryDis

Figure 4: Visualization of the t-SNE transformed embeddings of AutoDis, Binary Encoding (base-2), Novenary Encoding (base-9), and NaryDis. Points with similar colors have similar feature values. The embeddings of AutoDis retain continuity only. However, the results lack discriminability because the distances between different points in the two-dimensional space are too small. By contrast, the encoding-based solutions achieve more discriminative results.

performances. The improvement is especially obvious for DNN on the Criteo dataset, NaryDis improves AutoDis by 0.51% and 0.81% *w.r.t.* AUC and LogLoss, respectively.

4.3 In-depth Analysis

In this section, we further investigate NaryDis for a better understanding of the *continuity* and *discriminability*, as well as the hierarchical attention mechanism.

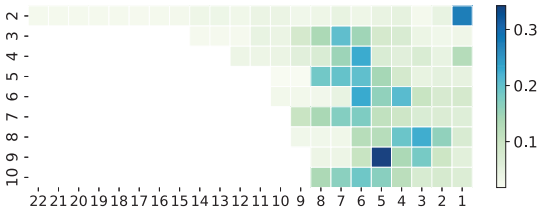
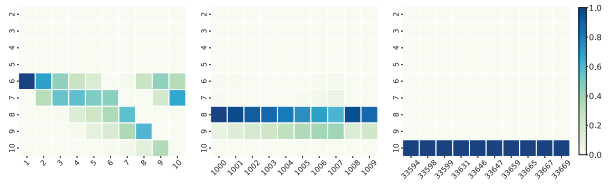


Figure 5: The intra-ary attention score of the 5-th numerical field (with the most features) with different n-ary encoding. The x-axis is the positions (right is low position) and the y-axis is n-ary encoding (from binary to decimal). The space in the lower left corner is unused positions.



(a) Small value (b) Medium value (c) Large value

Figure 6: The partial inter-ary attention score of the 5-th numerical field (with the most features) with different n-ary encoding. The x-axis is the feature values (with small, medium, and large values) and the y-axis is n-ary encoding (from binary to decimal).

4.3.1 Embedding analysis. To deeply investigate the properties of hybrid n-ary encoding-based discretization, we visualize the embeddings for AutoDis, binary encoding (base-2), novenary encoding (base-9), and NaryDis respectively. We first randomly select 250 embeddings that are derived from the 3-rd numerical field of the Criteo dataset as [10] does for a fair comparison and project them onto a two-dimensional space with t-SNE [33] as depicted in Figure 4. Despite continuous, the embeddings of AutoDis are deficient in discrimination, as the allocation probability is implicitly determined by a neural network. When adopting a single n-ary encoding, Figures 7(b) and 7(c) show an increase in discriminability. Interestingly, we note a pattern of clustering, and the size of each cluster grows with the base number used for encoding. In addition, continuity is also preserved in each cluster. The discontinuity among different clusters is caused by the two "*collision risk*" issues, *i.e.*, *carry* and *modulo congruence*. However, by assembling representations under hybrid n-ary encodings, NaryDis solves these issues and provides superior continuity and discriminability, which demonstrates the effectiveness of our proposed NaryDis framework.

4.3.2 Attention analysis. To investigate the hierarchical attention mechanism designed in NaryDis, we hence visualize the attention scores of the 5-th numerical field (with the most features) for intra-ary and inter-ary attention in Figure 5 and Figure 6, respectively. From Figure 5, we find that as the position increases, the intra-ary attention importance increases first and then decreases ($\curvearrowright$) generally. This phenomenon is slightly different from what we expected, that the attention importance increases ($\nearrow$) as the position increases, because higher positions weigh heavier in n-ary encoding. Our in-depth analysis finds that the hit rates for those high positions are relatively low due to the limited occurrence frequency of large feature values. Therefore, these large feature values with low frequency are insufficient to support the adequate training of

the corresponding parameters, so that the result is a compromise between importance and occurrence frequency.

In Figure 6, we show the inter-ary attention scores with small, medium, and large feature values. The figures show that the inter-ary attention usually favors larger base numbers (*e.g.*, $6 \sim 10$), especially for the features with larger values. The reason is that the "*collision risk*" occurs even less in the case of n-ary encoding with a larger base due to larger carry borders and positional values. Particularly, we are delighted to find that there exists a periodical pattern in the attention scores for each n-ary encoding. The reason is that the inter-ary attention learns to decline the weight of the n-ary encoding when approaching its carry borders and positional values, so that avoids the potential "*collision risks*".

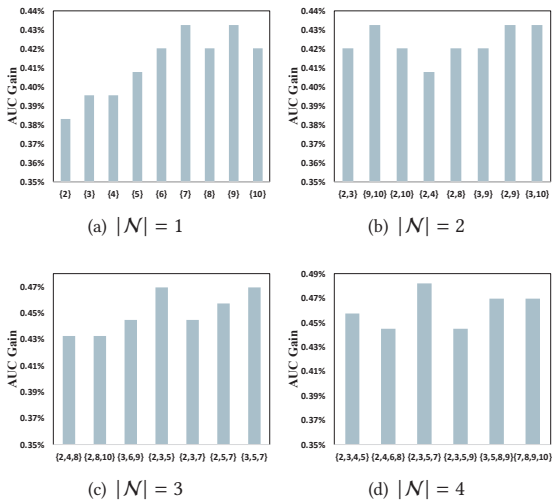
4.4 N-ary Selection

From Section 4.3.2, we can observe that the inter-ary attention network has a strong ability to choose the suitable encoding setting for each feature. Therefore, a naive solution is to provide a large encoding space $\mathcal{N}$ and let the inter-ary attention network decide which to choose. However, this solution will bring high overhead. To make NaryDis more efficient, we conduct experiments on the selection of encoding space $\mathcal{N}$ by applying NaryDis to the backbone DNN model. The size of encoding space varies from 1 to 4, and the experimental results on the Criteo dataset are shown in Figure 7. Several beneficial observations can be drawn from the results:

- NaryDis with a single n-ary encoding setting (*i.e.*, $|\mathcal{N}| = 1$) already outperforms the strongest baseline AutoDis. Besides, using a larger base (*e.g.*, $6 \sim 10$) can get a slightly better effect, which is consistent with the observation in Figure 6.
- Hybrid n-ary encoding achieves better performance than single n-ary encoding. By combining multiple n-ary, the potential "collision risks" can be circumvented cleverly, severing as a mixed-granularity discretization. Additionally, enlarging the encoding space $\mathcal{N}$ within a certain range is beneficial.
- Encoding space $\mathcal{N}$ consisting of prime numbers (*e.g.*, $2, 3, 5, 7, \ldots$) may contribute to easing "collision risks" and achieving better performance. This might be because the carry borders of prime-based encoding are distinct.

4.5 Model Complexity

In order to quantitatively analyze the time and space complexity of the NaryDis framework, we apply it to the backbone DNN model, and record the overall inference time (over the test dataset) and the model parameters on Criteo, as shown in Table 4. All the experiments are conducted on an NVIDIA Tesla V100-PCIe GPU with 32G memory. Specifically, compared to AutoDis, our single-ary vanilla model (with binary) slightly increases the parameters by 0.45%, while the inference time is reduced by 3.75% and the AUC already surpasses AutoDis significantly, which sufficiently demonstrates the superiority of our n-ary encoding-based solution. Besides, the increased inference time (+11.39%) and parameters (+4.17%) of hybrid-ary NaryDis (combining 2/3/5/7-ary encodings) are also acceptable in the industry. Remarkably, the AUC improvement (+0.48%) is very impressive in comparison with AutoDis.



(a) $|\mathcal{N}| = 1$ (b) $|\mathcal{N}| = 2$

(c) $|\mathcal{N}| = 3$ (d) $|\mathcal{N}| = 4$

Figure 7: The AUC gain of NaryDis compared with AutoDis under different encoding space $\mathcal{N}$.

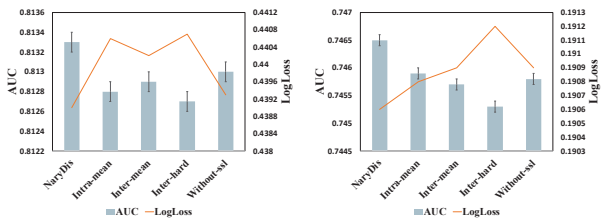
Table 4: Model complexity comparison. The improvement is computed over the strongest baseline AutoDis.

	AutoDis	EDD	NaryDis (vanilla)	NaryDis (2/3/5/7-ary)
Inference Time (s)	21.86	20.75	21.04	24.35
Relative Ratio	-	-5.08%	-3.75%	+11.39%
Parameters ($\times 10^7$)	2.012	2.009	2.021	2.096
Relative Ratio	-	-0.15%	+0.45%	+4.17%
AUC	0.8092	0.8074	0.8123	0.8131
Relative Ratio	-	-0.22%	+0.38%	+0.48%

4.6 Ablation Study and Hyper-Parameters Sensitivity

4.6.1 Ablation study. Three variants of NaryDis are devised to explore how different designed components influence the performance. In particular, we replace the intra-ary attention by a simple mean pooling, named as **Intra-mean**. Besides, the inter-ary attention is changed to mean pooling (soft-selection) and Gumbel-Softmax [16] (hard-selection), termed as **Inter-mean** and **Inter-hard**, respectively. We also remove the self-supervised regularization $\mathcal{L}_{ssl}$ from Eq.(10) and obtain the variant **Without-ssl**. The comparison of these variants with the original NaryDis is presented in Figure 8. We can observe that, after replacing the designed attention mechanism, the performance on both Criteo and AutoML drops, which emphasizes the significance of the hierarchical attention mechanism for capturing the correlation of both different positions within an encoded sequence and different n-ary encodings. Besides, performing a hard-selection strategy for choosing a single n-ary encoding is a sub-optimal approach. Additionally, removing the SSL affects the performance, confirming the benefit of self-supervised regularization. Our proposed NaryDis is consistently superior to all variants, showing that all the components together yield the best performance.

4.6.2 The impact of the α coefficient. To explore the impact of the self-supervised regularization coefficient α, we vary it in the range of $\{0, 0.1, 0.2, \ldots, 1.2\}$. The performance comparison on the Criteo



(a) Criteo. (b) AutoML.

Figure 8: Performances of different variants of NaryDis.

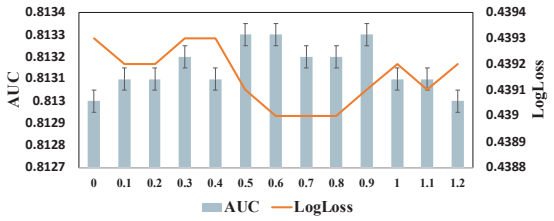


Figure 9: Impact of the α coefficient.

dataset is presented in Figure 9. The results show that the performance of NaryDis firstly grows and then slightly drops with the increase of α. Additionally, the optimal self-supervised regularization coefficient is achieved at around 0.5-0.9.

5 APPLICATION: INDUSTRIAL ADVERTISEMENT SYSTEM

5.1 System Overview

The overview of a mainstream large-scale advertisement platform is shown in Figure 10. The industrial advertisement system mainly consists of three core components: *Feature Factory Processing*, *Offline Training*, and *Online Serving*. Specifically, users' historical behaviors over the ads are collected and stored in the logs. Then these logs will be periodically processed in the feature factory, which contains several steps: 1) Feature extraction: extract the desired categorical and numerical features; 2) Feature filtering: filter the low-frequency features for avoiding feature cold-start; 3) Numerical feature discretization: discretize the numerical features into categorical features via pre-defined rules (only required for *hard-discretization* methods); 4) Feature encoding: leverage the original or hashed features to construct one-hot feature map; 5) Feature conversion: convert the instances into one-hot feature vectors according to the feature map.

When a user request arrives, the recall stage is first triggered and retrieves a list of candidate ads that are highly relevant to the user from the ads candidate pool. Then an indexer gathers the processed features and concatenates them into instances. A ranker leverages these instances and a CTR prediction model trained offline to estimate the click probability $Pr(y|\mathbf{x})$ for each candidate ad. Finally, a sorted ads list is generated based on some ranking function (*e.g.*, eCPM) and exposed to the user.

5.2 Performance Comparison

We collect and sample 8 consecutive days of user behavior records from a mainstream large-scale advertisement platform, where millions of daily active users generate tens of millions of user logs. The

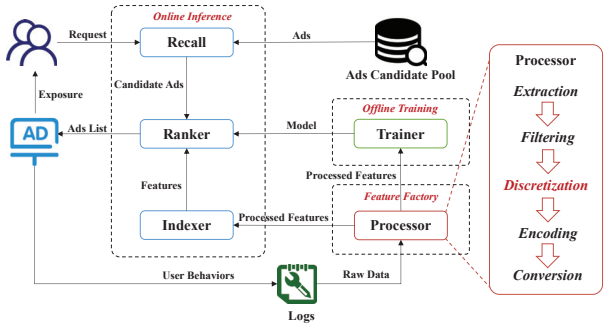


Figure 10: The overview of a large scale advertisement platform.

Table 5: The performance comparison in the large scale advertisement system.

Method	AUC	Logloss	Rel Improv.
FE	0.7248	0.1369	0%
EDD	0.7251	0.1371	0.04%
LD	0.7265	0.1368	0.23%
TD	0.7262	0.1369	0.19%
AutoDis	0.7281	0.1366	0.45%
NaryDis	**0.7298**	**0.1363**	**0.69%**

large-scale industrial dataset contains 41 numerical feature fields and 44 categorical feature fields. The numerical features include user statistics features (*e.g.*, number of ads clicked by users) and ad statistics features (*e.g.*, number of clicks on the ad). These features are either non-negative integers (*e.g.*, counts) or non-negative floating-point numbers (*e.g.*, proportions). We use a popular deep CTR prediction model DeepFM as the backbone model and compare NaryDis with some numerical feature presentation methods shown in Section 4.1.3, including FE, EDD, LD, TD, and AutoDis.

The performance comparison is presented in Table 5, from which we can observe that NaryDis outperform other baselines by a significant margin. In comparison with the hard-discretization methods (*i.e.*, EDD, LD, and TD) that are widely-used in industry, NaryDis achieves remarkable AUC improvement ranging from 0.4% to 0.6%. Besides, NaryDis saves the discretization step in feature factory processing, thus avoiding the cost of manual discretization. Moreover, compared with the strongest baseline AutoDis, NaryDis upgrades the AUC by 0.24%, which is statistically significant.

6 CONCLUSION

In this paper, we propose NaryDis, a pluggable discretization and representation learning framework for numerical features based on the hybrid n-ary encoding, which ensures prominent *continuity* and *discriminability*. NaryDis first leverages hybrid n-ary encoding as an automatic discretization module to generate multiple sequences in a hybrid-grained manner. Then, hierarchical intra-ary and inter-ary attention are leveraged to discriminate the importance of different positions and n-ary encodings, respectively. Moreover, an auxiliary contrastive regularization module is designed to provide self-supervised constraint signals. Extensive experiments demonstrate the effectiveness and compatibility of NaryDis.

REFERENCES

[1] Bhaskar Bhattacharya and Desale Habtzghi. 2002. Median of the p value under the alternative hypothesis. *The American Statistician* 56, 3 (2002), 202–206.

[2] Bo Chen, Yichao Wang, Zhirong Liu, Ruiming Tang, Wei Guo, Hongkun Zheng, Weiwei Yao, Muyu Zhang, and Xiuqiang He. 2021. Enhancing Explicit and Implicit Feature Interactions via Information Sharing for Parallel Deep CTR Models. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 3757–3766.

[3] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *Proceedings of International conference on machine learning*. PMLR, 1597–1607.

[4] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. 2016. Wide & Deep Learning for Recommender Systems. In *Proceedings Workshop Deep Learning for Recommender Systems*. 7–10.

[5] Stephen Chrisomalis. 2010. *Numerical notation: A comparative history*. Cambridge University Press.

[6] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *Proceedings of the 10th ACM conference on recommender systems*. 191–198.

[7] Krzysztof Grabczewski and Norbert Jankowski. 2005. Feature Selection with Decision Tree Criterion. In *Proceedings of Fifth International Conference on Hybrid Intelligent Systems (HIS'05)*. IEEE Computer Society, 212–217.

[8] Thore Graepel, Joaquin Quinonero Candela, Thomas Borchert, and Ralf Herbrich. 2010. Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft's bing search engine. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*. 13–20.

[9] Yulong Gu, Zhuoye Ding, Shuaiqiang Wang, Lixin Zou, Yiding Liu, and Dawei Yin. 2020. Deep Multifaceted Transformers for Multi-objective Ranking in Large-Scale E-commerce Recommender Systems. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2493–2500.

[10] Huifeng Guo, Bo Chen, Ruiming Tang, Weinan Zhang, Zhenguo Li, and Xiuqiang He. 2021. An Embedding Learning Framework for Numerical Features in CTR Prediction. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2910–2918.

[11] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. 1725–1731.

[12] Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 297–304.

[13] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 9729–9738.

[14] 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 *Proceedings of the eighth international workshop on data mining for online advertising*. ACM, 5:1–5:9.

[15] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*. PMLR, 448–456.

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

[17] Manas R Joglekar, Cong Li, Mei Chen, Taibai Xu, Xiaoming Wang, Jay K Adams, Pranav Khaitan, Jiahui Liu, and Quoc V Le. 2020. Neural input search for large scale recommendation models. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2387–2397.

[18] Yu-Chin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. 2016. Field-aware Factorization Machines for CTR Prediction. In *Proceedings of the 10th ACM conference on recommender systems*. 43–50.

[19] Guolin Ke, Zhenhui Xu, Jia Zhang, Jiang Bian, and Tie-Yan Liu. 2019. DeepGBM: A Deep Learning Framework Distilled by GBDT for Online Prediction Tasks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 384–394.

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

[21] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[22] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. 2018. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1930–1939.

[23] 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 *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. 1222–1230.

[24] Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang, Udit Gupta, Carole-Jean Wu, Alisson G. Azzolini, Dmytro Dzhulgakov, Andrey Mallevich, Ilia Cherniavskii, Yinghai Lu, Raghuraman Krishnamoorthi, Ansha Yu, Volodymyr Kondratenko, Stephanie Pereira, Xianjie Chen, Wenlin Chen, Vijay Rao, Bill Jia, Liang Xiong, and Misha Smelyanskiy. 2019. Deep Learning Recommendation Model for Personalization and Recommendation Systems. *CoRR* abs/1906.00091 (2019).

[25] Yanru Qu, Bohui Fang, Weinan Zhang, Ruiming Tang, Minzhe Niu, Huifeng Guo, Yong Yu, and Xiuqiang He. 2018. Product-based neural networks for user response prediction over multi-field categorical data. *ACM Transactions on Information Systems (TOIS)* 37, 1 (2018), 1–35.

[26] Kan Ren, Weinan Zhang, Yifei Rong, Haifeng Zhang, Yong Yu, and Jun Wang. 2016. User response learning for directly optimizing campaign performance in display advertising. In *Proceedings of the 25th acm international on conference on information and knowledge management*. 679–688.

[27] Steffen Rendle. 2010. Factorization machines. In *2010 IEEE International conference on data mining*. IEEE, 995–1000.

[28] V Thomas Rhyne. 1970. Serial binary-to-decimal and decimal-to-binary conversion. *IEEE Trans. Comput.* 100, 9 (1970), 808–812.

[29] Matthew Richardson, Ewa Dominowska, and Robert Ragno. 2007. Predicting clicks: estimating the click-through rate for new ads. In *Proceedings of the 16th international conference on World Wide Web*. 521–530.

[30] Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. 2019. Autoint: Automatic feature interaction learning via self-attentive neural networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 1161–1170.

[31] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15, 1 (2014), 1929–1958.

[32] Yixin Su, Rui Zhang, Sarah Erfani, and Zhenghua Xu. 2021. Detecting beneficial feature interactions for recommender systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 4357–4365.

[33] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).

[34] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & Cross Network for Ad Click Predictions. In *Proceedings of the ADKDD'17*. ACM, 12:1–12:7.

[35] Ruoxi Wang, Rakesh Shivanna, Derek Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed Chi. 2021. DCN V2: Improved Deep & Cross Network and Practical Lessons for Web-scale Learning to Rank Systems. In *Proceedings of the Web Conference 2021*. 1785–1797.

[36] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. 2021. Self-supervised graph learning for recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 726–735.

[37] Xin Xin, Bo Chen, Xiangnan He, Dong Wang, Yue Ding, and Joemon Jose. 2019. CFM: Convolutional Factorization Machines for Context-Aware Recommendation.. In *IJCAI*, Vol. 19. 3926–3932.

[38] Tiansheng Yao, Xinyang Yi, Derek Zhiyuan Cheng, Felix Yu, Ting Chen, Aditya Menon, Lichan Hong, Ed H Chi, Steve Tjoa, Jieqi Kang, et al. 2020. Self-supervised Learning for Large-scale Item Recommendations. *arXiv preprint arXiv:2007.12865* (2020).

[39] Weinan Zhang, Tianming Du, and Jun Wang. 2016. Deep learning over multi-field categorical data. In *Proceedings of European conference on information retrieval*. Springer, 45–57.

[40] Weinan Zhang, Jiarui Qin, Wei Guo, Ruiming Tang, and Xiuqiang He. 2021. Deep Learning for Click-Through Rate Estimation. In *Proceedings of the International Joint Conference on Artificial Intelligence*. 4695–4703.

[41] Chenxu Zhu, Bo Chen, Weinan Zhang, Jincai Lai, Ruiming Tang, Xiuqiang He, Zhenguo Li, and Yong Yu. 2021. AIM: Automatic Interaction Machine for Click-Through Rate Prediction. *IEEE Transactions on Knowledge and Data Engineering* (2021).