Enhancing Interpretability and Effectiveness in Recommendation with Numerical Features via Learning to Contrast the Counterfactual samples

Xiaoxiao Xu
Kuaishou Technology
Beijing, China
xuxiaoxiao05@kuaishou.com

Hao Wu
Kuaishou Technology
Beijing, China
wuhao10@kuaishou.com

Wenhui Yu
Kuaishou Technology
Beijing, China
yuwenhui07@kuaishou.com

Lantao Hu
Kuaishou Technology
Beijing, China
hulantao@kuaishou.com

Peng Jiang
Kuaishou Technology
Beijing, China
jiangpeng@kuaishou.com

Kun Gai
Unaffiliated
Beijing, China
gai.kun@qq.com

ABSTRACT

We propose a general model-agnostic **C**ontrastive learning framework with **C**ounterfactual **S**amples **S**ynthesizing (CCSS) for modeling the monotonicity between the neural network output and numerical features which is critical for interpretability and effectiveness of recommender systems. CCSS models the monotonicity via a two-stage process: synthesizing counterfactual samples and contrasting the counterfactual samples. The two techniques are naturally integrated into a model-agnostic framework, forming an end-to-end training process. Abundant empirical tests are conducted on a publicly available dataset and a real industrial dataset, and the results well demonstrate the effectiveness of our proposed CCSS. Besides, CCSS has been deployed in our real large-scale industrial recommender, successfully serving over hundreds of millions users.

CCS CONCEPTS

• **Recommender systems**;

KEYWORDS

Recommender system, Numerical features, Monotonicity, CTR prediction

ACM Reference Format:
Xiaoxiao Xu, Hao Wu, Wenhui Yu, Lantao Hu, Peng Jiang, and Kun Gai. 2024. Enhancing Interpretability and Effectiveness in Recommendation with Numerical Features via Learning to Contrast the Counterfactual samples. In *Companion Proceedings of the ACM Web Conference 2024 (WWW '24 Companion), May 13–17, 2024, Singapore, Singapore.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3589335.3648345

WWW '24 Companion, May 13–17, 2024, Singapore, Singapore.

ACM ISBN 979-8-4007-0172-6/24/05…$15.00
https://doi.org/10.1145/3589335.3648345

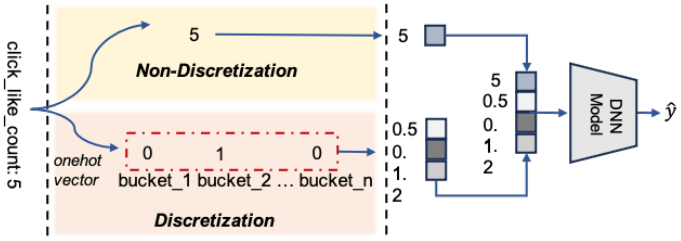

Figure 1: An illustration of the numerical features usage in recommender's deep model: utilized both as dense values by Non-discretization approach and as embedding by Discretization approach.

1 INTRODUCTION

Recommender systems equipped with deep models have been widely deployed industrially to alleviate the problem of information overload. Deep models in industrial recommender systems, eg., CTR prediction model and likes prediction model, adopt numerical features as well as categorical features as inputs. Numerical features learning has become an active research task in industrial recommender systems [4, 8]. Illustrating the activity levels of users, items and the interactions between specific users and items, numerical features are crucial to the performance of deep models. In practice, learning approach for numerical features can be categorized into two groups: (1) Non-discretization: using the original values or its transformations directly as dense inputs [2, 5, 12]; (2) Discretization: transforming continuous numerical features into categorical features through discretization strategy and assigning embedding like categorization strategy [8]. In general, the above two learning approaches for numerical features are both adopted in a single deep neural network, as illustrated in Figure 1.

Most of the existing research for numerical features focuses on discretization and representation techniques to retain continuity for similar features [4, 8], but the prior semantic relation, i.e., monotonicity between the neural network output and the numerical inputs has been seldom addressed publicly. The monotonicity

First Author and Second Author contribute equally to this work.

between the model output and numerical inputs is critical to the interpretability of the neural network. In an industrial Recommender system, with the same context, the higher the activity level of the interaction between the user and the item, the higher predicted score of the neural network. Especially in User-Generated Content based platforms, such as Tiktok, Kuaishou, and etc., the consumer expect to see UGC with high quality illustrated by numerical features. Besides, in UGC-based platforms, contents with high quality are rewarded by more chance to display, which is critical for motivating the producer. Besides, the monotonicity between the neural network outputs and numerical inputs is one kind of prior knowledge of the Recommender system, it is helpful for improving the accuracy of the model.

Traditional methods for modeling the monotonicity between neural network output and the numerical inputs are of two types: (1) only adopt numerical inputs as dense features and fuse them with DNN output linearly [10]; (2) only adopt numerical inputs as dense features with a delicately designed network to retain monotonicity [16]. Above methods both require that numerical features should only be utilized as dense values through Non-discretization approach. In a real industrial Recommender system, besides adopted as dense features by Non-discretization approach, numerical features are also learned as embedding by Discretization approach, as shown in figure 1, in which the traditional methods are no longer applicable.

To explicitly model the monotonicity between the neural network outputs and numerical inputs and make better use of this prior knowledge, we propose an adoptable model-agnostic **C**ontrastive learning framework with **C**ounterfactual **S**amples **S**ynthesizing (CCSS). CCSS consists one Counterfactual Samples Synthesizer and one Contrastive Objective Function Module which can cooperatively serve as a plug-and-play component to any neural network. The Counterfactual Synthesizer generates a counterfactual sample and a factual sample by disturbing the value of numerical features and keeping other features unchanged of the original sample. To synthesize generated samples as realistic as possible, which can take advantage of the contrastive loss to the greatest extent, we disturb only one numerical feature for each original sample. Moreover, to generate samples as informative as possible, we introduce feature importance as the specific metric to quantize the probability to be disturbed for each numerical feature. The contrastive objective function learns to rank the model outputs of the generated sample and the original sample, which enable the model to learn the monotonicity between model outputs and numerical features in an end-to-end manner. We summarize the main contributions as follows:

- As far as we know, we are the first to clarify the importance to neural network's interpretability and effectiveness of the monotonicity between the neural network output and the numerical features.
- We propose a general model-agnostic Contrastive learning framework with Counterfactual Samples Synthesizing(CCSS). The counterfactual and factual samples are generated locally based on the original samples, which enable CCSS to be adoptable in online learning scheme. The contrastive loss among the counterfactual sample, the factual sample and the original sample makes it possible to model the monotonicity end-to-end.
- We verify the effectiveness and adaptability of our proposed CCSS through extensive experiments conducted on a public benchmark dataset and a large scale industrial dataset, and successfully apply it in a real-world large scale recommender system bringing a considerable performance improvement.

2 RELATED WORK

In this section, we will introduce the related work from three aspects: Interpretability in Recommendation, Numerical Features Modeling in Recommendation and Counterfactual Sample Synthesizing.

2.1 Interpretability in Recommendation

Recommender systems play a crucial role in a wide range of web applications and services, in terms of distributing online contents to targeted users who are likely to be interested in them. The majority of the efforts in recommender system have been put into developing more effective model structures to achieve better performance, while the interpretability, i.e., explainability of recommender system has been seldom addressed. The interpretability of recommender system gives reasons to clarify why such results are derived [15], which benefits recommender system in to ways: 1) interpretability helps system maintainers to diagnose and refine the recommendation pipeline. 2) interpretability promote persuasiveness and customer satisfaction by increasing transparency [11]. Research on interpretability in recommendation can be grouped into two directions [21]: giving recommendation reasons by post-hoc methods [13] and building explainable models or add explainable components to give reasonable recommendation [11]. Recently, the efforts in recommender's interpretability have been concentrated on introducing more structural information as model inputs [17]. Our work falls into the second direction. In this paper, we propose to improve the interpretability of models in recommendation from the perspective of building the monotonicity between the outputs of the model and the numerical feature, in which the research is limited.

2.2 Numerical Features Modeling in Recommendation

In recommendation, e.g., CTR prediction model, learning of numerical features is one of the hot research topics. Most of the work is aimed to learning better embedding for numerical features in Discretization approach by means of addressing SBD (Similar value But Dis-similar embedding) and DBS (Dis-similar value But Same embedding) problems [4, 8, 9]. In this paper, we propose to learn the monotonicity between model outputs and numerical features, which is another type of prior information of numerical features. It is helpful for deriving more reasonable recommendation to model the prior monotonicity between neural network outputs and numerical features. To explain this, we take a frequently-used numerical feature *click_like_count* for instance. With other features being the same, the video with higher *click_like_count* should be ranked higher than the other in the same context. Our work model the

O	Original sample
$\mathcal{F}$	Factual sample
C	Counterfactual sample
E	embedding parameters
x_i^c	feature value of the i-th categorical feature field
x_j^n	feature value of the j-th numerical feature field

Table 1: Important notations.

monotonicity end-to-end with an model-agnostic method, which is the first attempt in numerical feature learning research domain.

2.3 Counterfactual Sample Synthesizing

Counterfactual samples have been recently used for data augmentation in Visual Question Answering (VQA) and Natural Language Processing (NLP). In VQA, counterfactual samples are synthesized by masking the critical objects in images or words in questions [1], which is much adoptable than traditional adversary-based data augmentation methods. In NLP, counterfactual samples are synthesized by removing phrases, which alThe generated counterfactual samples have also been used for explanations of deep learning, i.e. interpretability [19]. In recommendation, we synthesize counterfactual samples by directionally disturbing numerical feature values of the original samples. For each sample, the numerical feature to be disturbed is sampled with its feature importance as sampling probability. In addition, to model the monotonicity between the model output and numerical features, we introduce auxiliary contrastive objective ranking the model outputs of the counterfactual sample and the factual sample.

3 PROPOSED METHOD

In Section 3.1, we firstly review the background of deep models in industrial recommender systems, using CTR prediction as an example for illustration. Then we describe how to synthesize counterfactual samples in our proposed framework in Section 3.2. We further dig into the details of the implementations during training in Section 3.3. The notations are summarized in Table 1.

3.1 Preliminaries

Given an user, a candidate item and the contexts in an impression scenario, CTR prediction, is to infer the probability of a click event. The CTR prediction model is mostly formulated as a supervised logistic regression task and trained with an i.i.d. dataset $\mathcal{D}$ collected from historic impressions. Each instance $\boldsymbol{O} = (\boldsymbol{x}, y) \in \mathcal{D}$ contains the features $\boldsymbol{x}$ implying the information of $\{user, item, contexts\}$, and the label $y \in \{0, 1\}$ observed from user implicit feedback. $y = 1$ indicates an instance with positive label, and an instance with negative label is indicated by $y = 0$. The instance feature $\boldsymbol{x}$ is a multi-field data record including M numerical fields and N categorical fields:

$$\boldsymbol{x} = concatenate([x_1^c, x_2^c, ..., x_M^c], [x_1^n, x_2^n, ..., x_N^n]) \tag{1}$$

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

$$\boldsymbol{e}_i^c = \boldsymbol{E}_i^c \cdot onehot(x_i^c) \tag{2}$$

where $\boldsymbol{E}_i^c \in \mathbb{R}^{v_i \times d}$ is the embedding matrix or i-th categorical field, v_i and d is the vocabulary size and embedding size. Numerical features are used both as dense features by Non-discretization approach and embedding by Discretization approach. For the j-th numerical feature field, the feature embedding can be obtained by a two-stage operation: discretization and embedding look-up.

$$\boldsymbol{e}_j^n = \boldsymbol{E}_j^n \cdot onehot(d_j(x_i^n)) \tag{3}$$

where $\boldsymbol{E}_j^n \in \mathbb{R}^{w_i \times d}$ is the learnable embedding matrix or j-th numerical field, w_i is the number of the buckets after discretization. Then all the embedding of categorical features and numerical features, as well as dense values of numerical features are concatenated to form the input of MLP Module afterwards, i.e,

$$\begin{aligned} \boldsymbol{z} = concatenate([&\boldsymbol{e}_1^c, \boldsymbol{e}_2^c, ..., \boldsymbol{e}_M^c, \\ &\boldsymbol{e}_1^n, \boldsymbol{e}_2^n, ..., \boldsymbol{e}_N^n, \\ &[x_1^n, x_2^n, ..., x_N^n]]) \end{aligned} \tag{4}$$

Subsequently, the estimated CTR $\hat{y}$ can be obtained by the following discriminative model,

$$\hat{y} = \sigma(f_{\boldsymbol{\theta}}(\boldsymbol{z})) \tag{5}$$

where $f_{\boldsymbol{\theta}}(\cdot)$ refers to the function of the MLP Module which is parameterized by $\boldsymbol{\theta}$, and $\sigma(\cdot)$ is the sigmoid activation function. The model parameters $\boldsymbol{E}$ and $\boldsymbol{\theta}$ are learned by maximizing the objective function $\mathcal{L}(\boldsymbol{E}, \boldsymbol{\theta})$ with gradient-based optimization methods. In the traditional point estimate CTR prediction model, the objective function is equal to the negative log-likelihood $l(\boldsymbol{E}, \boldsymbol{\theta})$:

$$\begin{aligned} \mathcal{L}(\boldsymbol{E}, \boldsymbol{\theta}) &= l(\boldsymbol{E}, \boldsymbol{\theta}) \\ &\equiv -y\log\hat{y} - (1-y)\log(1-\hat{y}) \end{aligned} \tag{6}$$

3.2 Counterfactual Sample Synthesizing

To take advantage of the contrastive loss to the greatest extent, we synthesize generated samples as realistic as possible, i.e., the difference between generated samples and the original samples are as small as possible. Therefore, we disturb only one numerical feature for each original sample. To generate samples as informative as possible, we introduce feature importance as the specific metric to quantize the probability to be disturbed for each numerical feature. In this chapter, we describe the implementation details to synthesize realistic and informative generated samples and how to train with them, which is illustrated in Figure 2.

3.2.1 Interpreting Feature Importance. In machine learning, feature importance is a specific metric to measure the marginal contribution of each feature to model's decisions. When the model's decisions are affected much stronger by a specific feature, the stronger interpretability between our model's outputs and this specific feature is expected. To try to conform the expectation, we introduce feature importance during synthesizing counterfactual samples. We interpret feature importance of each numerical feature using Shapley Value, which is one of the most widely adopted measures of feature importance as it has a solid theoretical foundation [7, 18].

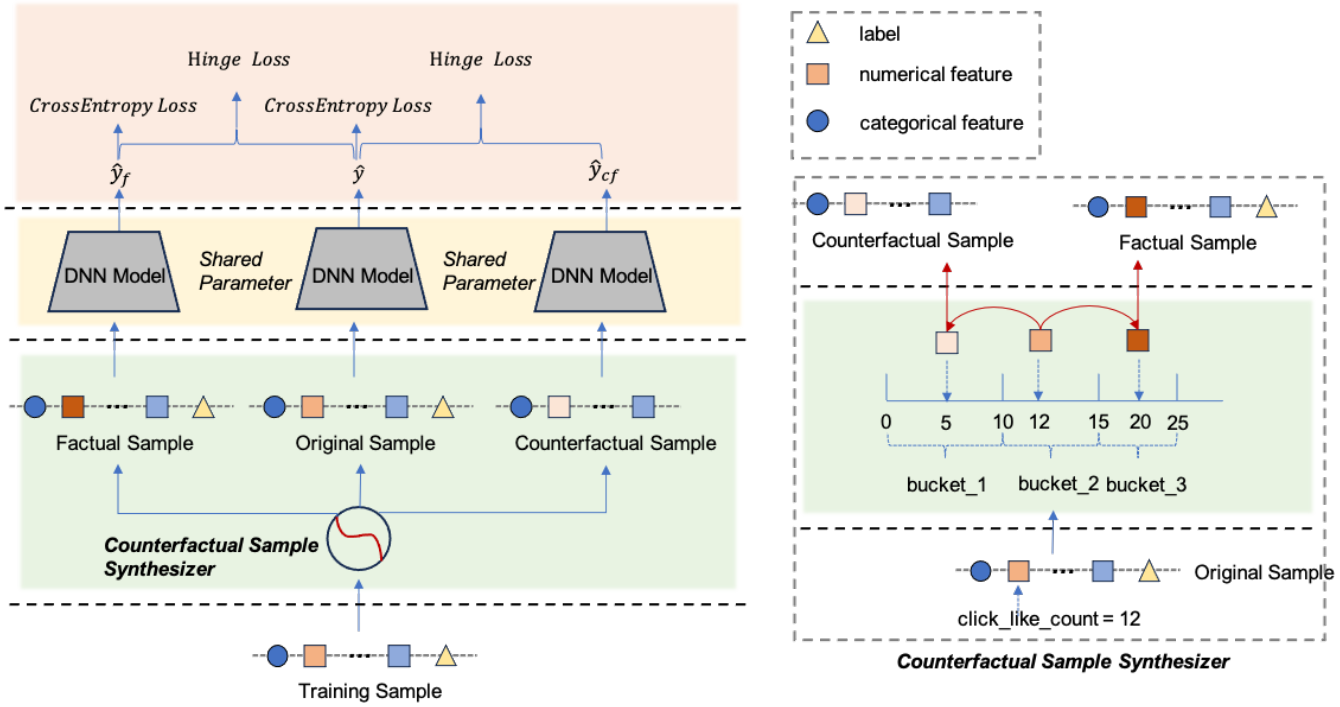

Figure 2: An illustration of our proposed CCSS: the left is the end-to-end learning framework with counterfactual sample synthesizing, and the right is the detail illustration of the Counterfactual Sample Synthesizer.

3.2.2 Feature Value Disturbing. We synthesize one counterfactual sample and one factual sample by disturbing one of the numerical features of each original samples. The synthesizing process can be divided into the following steps:

1) Obtain the feature importance q_i for each numerical feature where i is the index of the numerical feature.

2) Calculate the probability $p_i = \frac{q_i}{\sum_{j=1}^{N} q_j}$ to be disturbed for each numerical feature, where i is the index of the numerical feature.

3) Given a training sample, choose the index of the numerical feature to be disturbed by drawing floats from the uniform distribution over $[0, 1)$.

4) For an original training sample with positive label, we synthesize one counterfactual sample C by disturbing the feature value to the center of the left neighbor bucket while keeping other features unchanged, and the counterfactual sample has no known label. At the same time, we synthesize one factual sample $\mathcal{F}$ by disturbing the feature value to the center of the right neighbor bucket while keeping other features unchanged, and the factual sample has positive label. On the contrary, for an original training sample O with negative label, one counterfactual sample without known label is synthesized by disturbing the feature value to the center of the right neighbor bucket, and one factual sample with negative sample is synthesized by disturbing the feature value to the center of the left neighbor bucket. Here, we take the monotonic increasing relationship between model output and numerical inputs as an illustration. Table 2 demonstrates the monotonic decreasing scenario in detail.

Table 2: An detailed illustration of counterfactual samples synthesizing.

Monotonicity	Label	Disturb Destination	
		Counterfactual	Factual
Increasing	Positive	Right	Left
	Negative	Left	Right
Decreasing	Positive	Left	Right
	Negative	Right	Left

The Boundary Condition. For an original training sample with positive label, when it's feature value to be disturbed locates in the far-right bucket, we only synthesize one counterfactual sample by disturbing the feature value to the left neighbor bucket. Similarly, for an original training sample with negative label, when it's feature value to be disturbed locates in the far-left bucket, we only synthesize one counterfactual sample by disturbing the feature value to the right neighbor bucket.

3.3 Training with Counterfactual Samples

In this chapter, we describe the training implementation details with the synthesized samples and the original samples, which is illustrated in Figure 2.

3.3.1 Learning to Contrast. Here, we also take the monotonic increasing relationship between model output and numerical inputs as an illustration. For a training instance with positive label, the monotonicity means that we expect that its synthesized factual

sample $\mathcal{F}$ is scored higher by the model than this original sample O, and this original sample is scored higher than its synthesized counterfactual sample C. To address the monotonicity expectation in the CTR prediction model, we add pairwise losses to contrast the $(\mathcal{F}, O)$ pair and the (O, C) pair:

$$\begin{aligned}\mathcal{L}(E, \theta) &= l(E, \theta) \\ &+ \alpha(l^{\mathcal{P}}(E, \theta, \mathcal{F}, O) \\ &+ l^{\mathcal{P}}(E, \theta, O, C))\end{aligned} \tag{7}$$

where $l^{\mathcal{P}}$ denotes the pairwise loss, and hinge loss is adopted in this paper. we introduce a hyperparameter α to control the trade-off between pairwise loss and point-wise loss during training. Similarly, for a training instance with negative label, we expect that its synthesized counterfactual sample C is scored higher than this original sample O, and this original sample O is scored higher than its synthesized factual sample $\mathcal{F}$. Therefore, loss function for a training instance O with negative label is formulized as:

$$\begin{aligned}\mathcal{L}(E, \theta) &= l(E, \theta) \\ &+ \alpha(l^{\mathcal{P}}(E, \theta, O, \mathcal{F}) \\ &+ l^{\mathcal{P}}(E, \theta, C, O))\end{aligned} \tag{8}$$

3.3.2 Data Augmentation with Factual Samples. According to the priori monotonicity, the synthesized factual samples have known labels. For example, with the priori increasing monotonicity, a synthesized factual sample for a training instance with positive label is known to be labeled positive, while to be labeled negative for a training instance with negative label. In this paper, we exploit these synthesized factual sample with known labels to augment training data. Hence, our loss function can be rewritten as:

$$\begin{aligned}\mathcal{L}(E, \theta) &= l(E, \theta, O) \\ &+ l(E, \theta, \mathcal{F}) \\ &+ \alpha(l^{\mathcal{P}}(E, \theta, O, \mathcal{F}) \\ &+ l^{\mathcal{P}}(E, \theta, C, O))\end{aligned} \tag{9}$$

4 EXPERIMENTS

In this section, we conduct experiments with the aim of answering the following three research questions:

- **RQ1** How does CCSS preform for improving the interpretability and effectiveness of deep models in recommender?
- **RQ2** How does CCSS perform when plugged into various network backbones?
- **RQ3** What are the effects of the random strategy for selecting feature to be disturbed, constrastive loss, data augmentation loss and hyper-parameter α in CCSS?

4.1 Dataset

We evaluate the performance of our proposed approaches on a public benchmark dataset and our real industrial dataset.

KuaiRand-Pure: This data set is collected from the recommendation logs of the video-sharing mobile app, Kuaishou. It contains numerous numerical features indicating the statistical information of user behaviors and video behaviors. We adopt *is_click* as the binary label and adopt its full feature set for training and testing. Samples before 05-01 are used for training, and samples during 05-01 and 05-08 are used for testing.

Real Industrial Dataset: This data set is sampled and collected from the realtime data stream of our online video-sharing platform. It consists of 1.29 billion video play records generated by 43.98 million users within 8 days. Features in this dataset contain user ID, video ID and 14 dimensional numerical features which contain rich user activity level, video popularity and user-item interaction activity level information. The feature description in our real industrial dataset is shown in Table 3. Each instance in this dataset contains a binary label *is_collect* indicates whether the user collect this video to her favorites list in this instance. Our experiments are conducted on a binary classification model to predict *is_collect*. Samples from the beginning 7 days are used for training, and samples from the following 8-th day are used for testing. The statistics of the real industrial dataset can be found in Table 4.

For both KuaiRand and our real industrial dataset, there is a strong semantic correlation between the input numerical features and the estimated target, and we expect that "the larger the input value, the larger the output result should be".

Table 3: Feature description for our industrial dataset.

Feature Types	Descriptions
User ID	the unique identification of the user.
Video ID	the unique identification of the video.
video_ctr, video_ltr, video_wtr, video_ftr, video_vtr, video_lvtr, video_cmtr	global empirical click rate, like rate, follow rate, forward rate, video viewing duration, longview rate, comment rate of this video during 30days.
user_ctr, user_ltr, user_wtr, user_ftr, user_vtr, user_lvtr, user_cmtr	global empirical click rate, like rate, follow rate, forward rate, video viewing duration, longview rate, comment rate of this user on this video's category during 30days.

Table 4: Statistics of the training dataset.

#user	43.98 Mil.
#video	23.80 Mil.
#training sample	1.29 Bil.
#test sample	0.26 Bil.

https://github.com/chongminggao/KuaiRand

Table 5: Model comparison on KuaiRand and our industrial dataset. We record the mean results over 5 runs. Std ≈ 0.1%, extremely statistically significant under unpaired t-test. * indicates the improvement is statistically significant at the significance level of 0.05 over the best baseline on AUC or GAUC. '$\mathcal{K}$th_fea' is short for 'Mono_rate of the numerical feature who has the $\mathcal{K}$th highest feature importance score'.

	Models	AUC	RelaImpr	GAUC	RelaImpr	1st_fea	2nd_fea	3rd_fea	4th_fea	5th_fea	6th_fea	7th_fea
Our Industrial Dataset	DNN	0.7200	-	0.6560	-	51.3%	59.8%	53.7%	53.9%	54.5%	55.0%	59.6%
	DNN +CCSS	0.7860*	+30.0%	0.7320*	+48.7%	83.6%	91.7%	79.2%	65.7%	82.7%	83.9%	87.3%
	Wide & Deep	0.7311	-	0.6672	-	54.5%	63.9%	55.8%	51.7%	50.1%	45.7%	41.0%
	Wide & Deep +CCSS	0.7763*	+19.6%	0.7110*	+26.2%	87.5%	89.1%	82.0%	76.4%	74.9%	87.3%	78.9%
	PNN	0.7253	-	0.6662	-	52.6%	54.1%	56.9%	58.1%	48.2%	52.9%	55.7%
	PNN +CCSS	0.7663*	+18.2%	0.6934*	+16.4%	88.0%	83.9%	80.3%	78.9%	70.2%	74.7%	87.9%
	DCN	0.7356	-	0.6702	-	53.7%	56.5%	50.4%	46.7%	47.8%	47.5%	49.5%
	DCN +CCSS	0.7865*	+21.6%	0.7242*	+31.7%	87.9%	90.2%	88.9%	88.0%	78.2%	87.5%	85.0%
	DeepFM	0.7168	-	0.6545	-	42.7%	45.6%	53.2%	53.6%	48.7%	53.8%	51.6%
	DeepFM +CCSS	0.7298*	+6.0%	0.6615*	+4.5%	81.1%	83.2%	80.8%	73.5%	74.0%	87.9%	84.5%
	Models	AUC	RelaImpr	GAUC	RelaImpr	1st_fea	2nd_fea	3rd_fea	4th_fea	5th_fea	6th_fea	7th_fea
KuaiRand-Pure	DNN	0.6847	-	0.6419	-	60.3%	58.2%	49.7%	50.9%	54.5%	50.2%	40.5%
	DNN +CCSS	0.7324*	+25.8%	0.6768*	+24.5%	66.5%	62.1%	65.1%	71.3%	80.5%	66.7%	75.8%
	Wide & Deep	0.7113	-	0.6626	-	61.5%	59.7%	51.3%	51.5%	57.8%	51.1%	45.7%
	Wide & Deep +CCSS	0.7527*	+19.6%	0.6959*	+20.5%	68.0%	65.4%	64.6%	72.7%	88.3%	68.8%	75.2%
	PNN	0.7222	-	0.6702	-	63.6%	60.9%	52.2%	52.0%	55.5%	50.6%	46.4%
	PNN +CCSS	0.7491*	+12.1%	0.7039*	+19.8%	68.6%	74.7%	63.3%	79.2%	87.3%	70.8%	77.5%
	DCN	0.7086	-	0.6605	-	59.3%	59.2%	49.7%	55.9%	53.5%	52.2%	47.5%
	DCN +CCSS	0.7737*	+31.2%	0.6873*	+16.7%	74.5%	75.8%	70.1%	78.0%	78.3%	73.2%	80.3%
	DeepFM	0.6912	-	0.6476	-	60.8%	58.4%	51.5%	51.1%	52.3%	51.8%	46.9%
	DeepFM +CCSS	0.7357*	+23.3%	0.6846*	+25.1%	65.8%	62.7%	67.2%	78.9%	78.7%	69.1%	78.6%

4.2 Baselines

In order to prove the plug-and-play nature of the method, we did sufficient tests on different backbones:

(1) DNN is the lightest model structure which only contains a MLP and some nonlinear activation functions.

(2) Wide & Deep [3] develop wide linear models and deep neural networks together to enhance their respective abilities.

(3) DeepFM [9] is a deep recommendation method that learns both low and high level interactions between fields.

(4) DCN [20] is based on DNN, explicitly applies feature crossing at each layer, eliminating the need for human feature engineering.

(5) PNN [14] employs a feature extractor to investigate feature interactions among inter-field categories.

4.3 Experimental Settings

In this chapter, we describe the implementation details and the evaluation metrics in our experiments.

4.3.1 Implementation Details. We utilize the same model settings for all approaches on each dataset to provide a fair comparison. For all the three datasets, we fix embedding size as 32 and DNN as 4 FC layers with [512, 255, 127, 127] hidden units. Furthermore, for DCN, we set the number of cross layer to 3. We optimize all approaches using mini-batch Adam, where the learning rate is 0.05 and decay_rate is 0.9. Furthermore, the batch size of all models is set to 1024. We adopt hyper-parameter $\alpha = 1.0$ for all the offline and online evaluations.

4.3.2 Evaluation Metrics. We adopt **AUC** and **Mono_rate** to evaluate the effectiveness and interpretability of our model.

AUC: [6] is a common metric for recommendation [20, 22]. It measures the goodness of order by ranking all the items with prediction. Thus we adopt AUC as the main metric.

GAUC: gauc is introduced in [22] which measures the goodness of intra-user order by averaging AUC over users and is shown to be more relevant to online performance in recommendation system. We adapt this metric in our experiments, gauc is defined as:

$$GAUC = \frac{\sum_{i=1}^{n} \#impression_i * AUC_i}{\sum_{i=1}^{n} \#impression_i} \tag{10}$$

where n is the number of users, $\#impression_i$ and AUC_i are the number of impressions and AUC corresponding to the i-th user.

In addition, we follow [20, 22] to introduce RelaImpr metric to measure relative improvement over models. For a random guesser, the value of AUC or GAUC is 0.5. Hence, RelaImpr is defined as:

$$RelaImpr(AUC) = \left(\frac{AUC(measured\ model) - 0.5}{AUC(base\ model) - 0.5} - 1\right) \times 100\% \tag{11}$$

$$RelaImpr(GAUC) = (\frac{GAUC(measured\ model) - 0.5}{GAUC(base\ model) - 0.5} - 1) \times 100\% \tag{12}$$

Mono_rate: We define Mono_rate to evaluate the monotonicity (i.e., interpretability) between model output and numerical features:

$$Mono\_Rate = \frac{\#Monotone\_pairs(D)}{\#Comparable\_pairs(D)} \tag{13}$$

where $Comparable\_pairs(D)$ donates the entire set of the expected monotonic pairs, including all the $(\mathcal{F}, O)$ pairs and (C, O) pairs. $Monotone\_pairs(D)$ donates the set of valid predicted monotonic pairs according to model outputs. For an original sample with positive label, the valid predicted monotonic pairs include $(\mathcal{F}, O)$ pairs whose $\hat{y}_f > \hat{y}$ and (C, O) pairs whose $\hat{y}_c f < \hat{y}$. Instead, for an original sample with negative sample, the valid predicted monotonic pairs include $(\mathcal{F}, O)$ pairs whose $\hat{y}_f < \hat{y}$ and (C, O) pairs whose $\hat{y}_{cf} > \hat{y}$.

4.4 Comparison with Baselines (RQ1, RQ2)

4.4.1 Offline Evaluation. To demonstrate the effectiveness and adaptability of the our proposed CCSS, we plug CCSS in many representative networks in binary classification task, such as DNN, Wide&Deep, DCN, DeepFM and PNN. According to Table 5, CCSS could improve all of the baselines on two large scale dataset. CCSS brings at least 6.0% AUC improvement and 4.5% GAUC improvement. To evaluate the effectiveness of CCSS to improve the monotonicity between model output and the numerical features during infer, we calculate Mono_rates with testing dataset. For simplicity, we calculate and compare the Mono_rates of the top 7 important numerical features on KuaiRand and our industrial dataset. According to Table 5, Mono_rate improvements are clearly remarkable, which means the interpretability of the neural network can be enhanced by our proposed CCSS.

4.4.2 Online Evaluation. To exam the effectiveness of CCSS in a real industrial recommendation scenario, we conduct online A/B testing based on a *is_collect* predicting model. Table 6 shows the experimental results. CCSS contributes 3.93% *collect_rate* gain when plugged into DCN which has already been highly optimized for our online system with abundant of features.

Table 6: Online A/B Testing.

Online A/B Test	collect_rate gain
DCN	-
DCN + CCSS	3.93%

4.5 Ablation Study(RQ3)

In this section, we demonstrate the advantages of our proposed random strategy for selecting feature to be disturbed, constrastive loss, data augmentation loss in CCSS. We present an ablation study on our real industrial dataset by evaluation several models based on DNN which is the lightest structures: (1) CCSS(Only Data Augmentation): only adopt the auxiliary point-wise loss of factual samples.

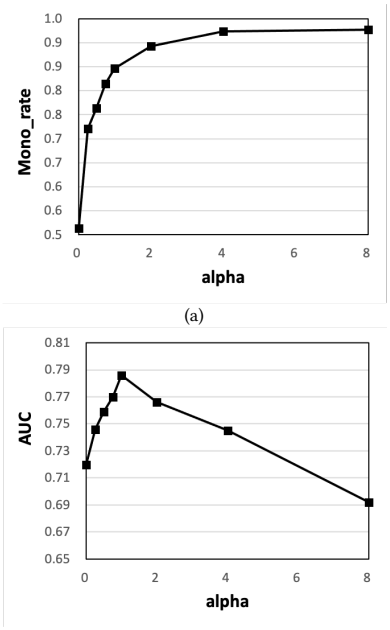

Figure 3: Effect of α on (a) Mono_rate and (b) AUC. DNN is used as the network.

(2) CCSS(Only Factual Contrastive Loss): only adopt the auxiliary hinge loss of Factual-Original sample pairs. (3) CCSS(Only Counterfactual Contrastive Loss): only adopt the auxiliary hinge loss of Counterfactual-Original sample pairs. (3) CCSS(Equal Probability Random Strategy): select the numerical feature to be disturbed with equal probability. The mean AUC and GAUC results over 5 runs are reported in Table 7, and the results confirm the advantages of our proposed random strategy for selecting feature to be disturbed, constrastive loss, data augmentation loss in CCSS.

Effect of HyperParameter α. We evaluate the effect of hyperparameter α which controls the trade-off between pairwise loss and point-wise loss. According to Figure 3, contrastive losses aimed to improve the interpretability of the model, can help to promote the effectiveness of the model with an appropriate proportion.

Table 7: Ablation study. The averaged AUC and GAUC results over 5 runs are reported.

Models	AUC	GAUC
DNN	0.786	0.732
DNN(Only Factual Pairwise loss)	0.772	0.715
DNN(Only Counterfactual Pairwise loss)	0.768	0.720
DNN(Equal Probability Random Disturb)	0.766	0.711
DNN(Only Factual Pointwise loss)	0.734	0.669

5 CONCLUSION

In this paper, we propose a general model-agnostic Contrastive learning framework with Counterfactual Samples Synthesizing (CCSS) for modeling the monotonicity between the neural network output and numerical features. CCSS introduces and models the monotonicity by synthesizing counterfactual sample and factual samples and learning to contrast among the counterfactual sample, factual samples and original samples. Besides, to generate samples as informative as possible, we introduce feature importance as the specific metric to quantize the probability to be disturbed for each numerical feature. Our proposed CCSS can be adapted to various representative networks without much effort and enjoys an end-to-end learning manner. Abundant offline and online experiments show that recommendation neural networks with CCSS can achieve better interpretability and performances.

REFERENCES

[1] Long Chen, Xin Yan, Jun Xiao, Hanwang Zhang, Shiliang Pu, and Yueting Zhuang. 2020. Counterfactual Samples Synthesizing for Robust Visual Question Answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

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

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

[4] Yuan Cheng. 2022. Dynamic Explicit Embedding Representation for Numerical Features in Deep CTR Prediction. In *Proceedings of the 31st ACM International Conference on Information Knowledge Management* (Atlanta, GA, USA) *(CIKM '22)*. Association for Computing Machinery, New York, NY, USA, 3888–3892. https://doi.org/10.1145/3511808.3557587

[5] 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.

[6] Tom Fawcett. 2006. An introduction to ROC analysis. *Pattern recognition letters* 27, 8 (2006), 861–874.

[7] Daniel Fryer, Inga Strümke, and Hien Nguyen. 2021. Shapley values for feature selection: The good, the bad, and the axioms. *Ieee Access* 9 (2021), 144352–144360.

[8] 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.

[9] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. *arXiv preprint arXiv:1703.04247* (2017).

[10] Malay Haldar, Prashant Ramanathan, Tyler Sax, Mustafa Abdool, Lanbo Zhang, Aamir Mansawala, Shulin Yang, Bradley Turnbull, and Junshuo Liao. 2020. Improving deep learning for airbnb search. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2822–2830.

[11] Ninghao Liu, Yong Ge, Li Li, Xia Hu, Rui Chen, and Soo-Hyun Choi. 2020. Explainable recommender systems via resolving learning representations. In *Proceedings of the 29th ACM international conference on information & knowledge management*. 895–904.

[12] Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang, Udit Gupta, Carole-Jean Wu, Alisson G Azzolini, et al. 2019. Deep learning recommendation model for personalization and recommendation systems. *arXiv preprint arXiv:1906.00091* (2019).

[13] Georgina Peake and Jun Wang. 2018. Explanation Mining: Post Hoc Interpretability of Latent Factor Models for Recommendation Systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery Data Mining* (London, United Kingdom) *(KDD '18)*. Association for Computing Machinery, New York, NY, USA, 2060–2069. https://doi.org/10.1145/3219819.3220072

[14] Yanru Qu, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, and Jun Wang. 2016. Product-based neural networks for user response prediction. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 1149–1154.

[15] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. " Why should i trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 1135–1144.

[16] Davor Runje and Sharath M Shankaranarayana. 2023. Constrained monotonic neural networks. In *International Conference on Machine Learning*. PMLR, 29338–29353.

[17] Ryotaro Shimizu, Megumi Matsutani, and Masayuki Goto. 2022. An explainable recommendation framework based on an improved knowledge graph attention network with massive volumes of side information. *Knowledge-Based Systems* 239 (2022), 107970. https://doi.org/10.1016/j.knosys.2021.107970

[18] Eunhye Song, Barry L Nelson, and Jeremy Staum. 2016. Shapley effects for global sensitivity analysis: Theory and computation. *SIAM/ASA Journal on Uncertainty Quantification* 4, 1 (2016), 1060–1083.

[19] Sahil Verma, Varich Boonsanong, Minh Hoang, Keegan E Hines, John P Dickerson, and Chirag Shah. 2020. Counterfactual explanations and algorithmic recourses for machine learning: A review. *arXiv preprint arXiv:2010.10596* (2020).

[20] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*. 1–7.

[21] Yongfeng Zhang, Xu Chen, et al. 2020. Explainable recommendation: A survey and new perspectives. *Foundations and Trends® in Information Retrieval* 14, 1 (2020), 1–101.

[22] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1059–1068.