Scenario-Adaptive Feature Interaction for Click-Through Rate Prediction

Erxue Min
erxue.min@gmail.com
Independent Researcher
Shenzhen, China

Da Luo
Kangyi Lin
Chunzhen Huang
lodaluo@tencent.com
plancklin@tencent.com
chunzhuang@tencent.com
Weixin Open Platform, Tencent
Guangzhou, China

Yang Liu*
yliukj@connect.ust.hk
The Hong Kong University of Science and Technology
Hong Kong, China

ABSTRACT

Traditional Click-Through Rate (CTR) prediction models are usually trained and deployed in a single scenario. However, large-scale commercial platforms usually contain multiple recommendation scenarios, the traffic characteristics of which may be significantly different. Recent studies have proved that learning a unified model to serve multiple scenarios is effective in improving the overall performance. However, most existing approaches suffer from various limitations respectively, such as insufficient distinction modeling, inefficiency with the increase of scenarios, and lack of interpretability. More importantly, as far as we know, none of existing Multi-Scenario Modeling approaches takes **explicit feature interaction** into consideration when modeling scenario distinctions, which limits the expressive power of the network and thus impairs the performance. In this paper, we propose a novel Scenario-Adaptive Feature Interaction framework named SATrans, which models scenario discrepancy as the distinction of patterns in feature correlations. Specifically, SATrans is built on a Transformer architecture to learn high-order feature interaction and involves the scenario information in the modeling of self-attention to capture distribution shifts across scenarios. We provide various implementations of our framework to boost the performance, and experiments on both public and industrial datasets show that SATrans 1) significantly outperforms existing state-of-the-art approaches for prediction, 2) is parameter-efficient as the space complexity grows marginally with the increase of scenarios, 3) offers good interpretability in both instance-level and scenario-level. We have deployed the model in WeChat Official Account Platform and have seen more than 2.84% online CTR increase on average in three major scenarios.

*Corresponding author.

KDD '23, August 6–10, 2023, Long Beach, CA, USA

ACM ISBN 979-8-4007-0103-0/23/08…$15.00
https://doi.org/10.1145/3580305.3599936

CCS CONCEPTS

• **Information systems** → **Computational advertising**; **Recommender systems**; • **Computing methodologies** → **Neural networks**.

KEYWORDS

Recommender Systems; Click-Through Rate Prediction; Multi-Scenario; Feature Interaction; Transformer

ACM Reference Format:
Erxue Min, Da Luo, Kangyi Lin, Chunzhen Huang, and Yang Liu. 2023. Scenario-Adaptive Feature Interaction for Click-Through Rate Prediction. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3580305.3599936

1 INTRODUCTION

In recent years, the Multi-Scenario Click-Through Rate (MS-CTR) Prediction [8, 19, 20, 28, 29], which focuses on predicting the CTR of user-item pairs in multiple scenarios, has become extensively explored for online recommendation. At large commercial companies like Tencent and Alibaba, there are often many business scenarios (e.g., home page feed, banner feed) [30], Besides, the log data collected from the service platform can be partitioned into multiple subsets according to some representative features (e.g., gender, country). These subsets have different CTR distributions and can be considered as scenarios [29]. Different scenarios share commonalities (e.g., overlapped users or items, general preferences) that can benefit the prediction in all scenarios. Meanwhile, user behaviors and exposure distributions can vary greatly across different scenarios [32]. Therefore, it is important to model both commonalities and distinctions among scenarios when estimating the CTR. Moreover, feature interaction[1] learning plays an essential role in the task of CTR prediction. Effectively modeling high-order combination of features can improve the expressive power of the network and thus contribute to better prediction performance [4, 10, 21].

There are generally three typical kinds of approaches for MS-CTR Prediction: **(1)** To leverage conventional CTR prediction models [4, 5, 10, 13, 24, 27] with heuristic training strategies, e.g, training a separate model for each scenario or training a shared model using

[1] In this paper, we use "feature interaction" and "feature combination" interchangeably as they are both used in the literature [4, 10, 21].

all scenario instances and then finetuning respectively. This category of methods can naturally inherit all merits of conventional CTR prediction models (e.g., explicit feature interaction), but their capability in knowledge transfer and speciality modeling across scenarios is limited. **(2)** To build a unified framework based on Multi-Task Learning (MTL) and treat each scenario as a task [2, 8, 20]. This strategy requires individual network modules (e.g., gate network, experts, or output tower) for each scenario, which consumes excessive parameters with the increase of scenarios. Worse still, MTL models normally assume the backbone network or expert network as a generalized Deep Neural Network (DNN) [11, 17, 22], which learns high-order feature interactions in a bit-wise[2] and implicit fashion, suffering from the insensitive gradient issue to discrete feature interaction and unable to fit a POLY function [14] or a simple dot product [16]. Although it is possible to replace DNN with some explicit interaction models such as Factorization Machines (FM) [15] or DCN [24], the process of feature interaction and scenario modeling is separated, which limits the model interpretability and might result in sub-optimal performance. **(3)** To generate Scenario-Adaptive Units (SAU) with an auxiliary encoder using scenario-specific features as input to influence the network [28–30]. These approaches are more flexible and parameter-efficient than MTL approaches to tackle a large number of scenarios and multiple scenario feature fields. However, existing approaches in this category do not directly and explicitly consider the impact of scenario specialities to feature interaction, so distinctions in feature correlations and combinations across scenarios are still unclear.

From the perspective of feature interaction, samples from different scenarios may have distinctive patterns. Taking e-commerce recommendation as an example, *gender*, *location* and *brand* could be three important features and their combinations may influence the CTR score significantly. However, the importance of the same feature combination varies in different scenarios. Considering second-order feature combinations, <*brand*, *location*> can be a more meaningful pair for food recommendation scenario as users' food preference is substantially influenced by geographical factors, while <*brand*, *gender*> might be more correlated in clothes recommendation due to the particular gender distinction in this scenario. As far as we know, none of the existing MS-CTR approaches can explicitly capture this kind of difference in feature interaction, which restricts the expressive power of the network and leads to the lack of model interpretability.

To address these limitations, in this paper, we propose **S**cenario-**A**daptive **Trans**former (**SATrans**), an explicit feature interaction model for MS-CTR Prediction, which involves the scenario information into the correlation modeling of features to learn distinctive and adaptive high-order feature interactions for each scenario. Concretely, we leverage Transformer [23] as the backbone architecture to model high-order interactions and combinations upon input features, which has been proved to be effective by AutoInt [21] and InterHAt [9]. The Multi-Head Self-Attention mechanism in Transformer allows each feature field to interact with all the others and automatically identify relevant features to form meaningful high-order features. To involve scenario specialities in the feature interaction, we first design a *Scenario Encoder* to transform scenario features into a fixed-length scenario embedding. The *Scenario-Adaptive Interacting Layers* are then leveraged to measure the correlations using both embeddings of the feature pair and the scenario embedding as input, in which the attention scores are calculated via a well-designed scenario-adaptive correlation function. The proposed scenario-adaptive self-attention mechanism entitles SATrans many merits: (1) **Commonality Modeling**: The shared feature transformation matrices in each interacting layer and the embedding layer naturally capture the common knowledge; (2) **Distinction Modeling**: The adaptive attention scores encode the distribution shifts across scenarios; (3) **High Scalability**: The scale of network parameters is marginally dependent on the number of scenarios, enabling SATrans to be efficient for thousands or even millions of scenarios; (4) **Good Explainability**: The attention scores can be a measure for feature correlations, which offers both instance-level and scenario-level explainability. To summarize, in this paper we make the following contributions:

- We are the first to model the MS-CTR Prediction problem from the feature interaction perspective and propose a novel SATrans which explicitly conducts scenario-adaptive high-order interaction over input features.
- We design three types of implementation for the scenario encoder and the scenario-adaptive interacting module for SATrans, respectively, which improve the feature interaction quality significantly compared with vanilla self-attention.
- We conducted extensive experiments on both public and industrial datasets. Experimental results on the task of multi-scenario CTR prediction show that our proposed approach not only outperforms existing state-of-the-art approaches remarkably in terms of prediction, but also possesses good scalability as well as model explainability.
- Considering the scarcity of open-source codes in MS-CTR Prediction, we publish the implementation of our model as well as the compared baselines[3] to facilitate future research.

2 RELATED WORK

2.1 Feature Interaction Modeling

In CTR prediction, it is difficult to achieve good results via directly using raw features or adding manually generated cross features [3, 4, 15]. Therefore, automatic feature interaction modeling has become a key role in this area to enhance the model expressive power [3, 4, 15, 24, 27]. An early attempt is Factorization Machines (FM) [15], which uses a low-dimensional vector to represent each feature field and learns second-order feature interaction through inner product, achieving a significant improvement over linear models. After that, further extensions of FM have been proposed. For example, Field-aware Factorization Machines (FFM) assign multiple independent embeddings for a feature to explicitly model feature interactions with different fields. In recent years, the success of deep learning also benefits CTR prediction performance. Wide&Deep [3] proposed by Google is one of the earliest published deep models

[2] Consider each element (bit) of the input vector as a unit, and perform interactions among the elements [25].

[3] https://github.com/qwerfdsaplking/SATrans

for CTR estimation, which adds the logit value of linear regression and DNN before feeding into the final sigmoid function. Moreover, DeepCross [18] extends residual network [5] for performing automatic feature interactions learning in an implicit way. However, as pointed out in [14, 16], it is more difficult for an MLP to effectively learn the high-order combining feature patterns than a dot product in FM, which suggests the explicit design of feature interaction learning in deep architectures, such as PNN [13], FNN [31], DeepFM [4], AFM [27], DCN [24], xDeepFM [10] and AutoInt [21]. Some of them [9, 10, 21, 24] are equipped with explicit high-order modeling capability [9, 10, 21, 24] and explainabiliy [9, 21]. However, these models are designed for single-scenario CTR prediction, and their effectiveness would be limited when the data is from multiple scenarios.

2.2 Multi-Scenario Modeling

Existing representative multi-scenario modeling approaches for CTR prediction can be categorized into three groups:

Heuristic training strategies: Conventional single-scenario CTR prediction models are trained with various heuristic strategies: (a) Training a separate model for each scenario, which models the speciality of each scenario individually but fails to capture the shared knowledge among multiple scenarios and requires a tremendous amount of computation and storage consumption for maintaining models; (b) Training a shared model with mixed samples from multiple scenarios without extra adjustment, which models the commonalities but neglects the distinctive traffic characteristics of different scenarios and thus cannot work well on all scenarios; (c) Training a base model with whole scenario samples and then fine-tuning model separately using samples from the corresponding scenario, which suffers from catastrophic forgetting issues [7] and also high complexity issue.

Multi-Task learning: Treating each scenario as a task, the MTL paradigm can be utilized to model the relationship between scenarios. SharedBottom [17] is the first MTL model with a hard parameter-sharing paradigm, which is simple but effective. MMOE [11] and PLE [22] adopt a soft parameter-sharing strategy named multi-gate mixture-of-experts [11], which utilizes different gate networks for each task to aggregate multiple experts among tasks. STAR [20] adopts the hard parameter-sharing paradigm as Shared Bottom [17] and designs a star topology framework, with one centred network to maintain whole-scenario commonalities and a set of scenario-specific networks to distinguish scenario distinctions. SAML [2] also uses individual branches of the network to model the speciality of each scenario and involves an auxiliary network to model the shared knowledge across all scenarios. HMoE [8] utilizes MMOE to implicitly model commonalities and distinctions among multiple scenarios. SAR-Net [19] is also based on a multi-expert architecture and introduces two scenario-aware attention modules to extract scenario-specific features on the user side. Inspired by casual intervention, CausalInt [26] designed an invariant representation modeling module for commonality extraction, negative effects mitigating module to retain specialities and inter-scenario transferring module to mitigate negative transfer.

Scenario-Adaptive units: To generate Scenario-Adaptive parameters or sub-networks using an auxiliary shared network. APG [28] use scenario-specific features and an auxiliary network to generate scenario-specific parameters for the entire prediction network. AdaSparse [29] proposes a generated scenario-aware pruner that produces neuron-level weighting factors to prune redundant neurons, which learns adaptive sparse structure for each scenario, achieving better generalization across scenarios with lower computational cost. M2M [30] proposes a meta attention module using network weights generated by specific scenario knowledge. As far as we know, none of the existing MS-CTR models take explicit feature interaction into consideration when modeling scenario distinctions, which limits their performance and explainability.

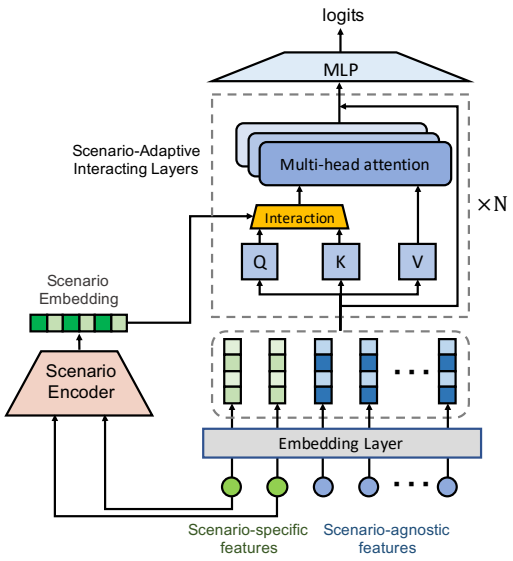


Figure 1: The overall framework of SATrans. The left part is the Scenario Encoder, generating fixed-size embeddings using scenario-specific features as input. The right part is the backbone network composed of multiple SAI layers. Implementation details of the Scenario Encoder and SAI layers are elaborated in Section 4.2 and 4.3, respectively.

3 PROBLEM FORMULATION

A dataset for CTR prediction can be represented as $\mathcal{D} = \{(x_j, y_j)\}_{j=1}^{|\mathcal{D}|}$, where x_j and $y_j \in \{0, 1\}$ denote the feature set and click label of the j-th sample. In real-world recommendation, there are often multiple business scenarios, which means the dataset $\mathcal{D}$ can be partitioned into multiple scenario-specific subsets $\mathcal{D}^s$ (i.e., $\mathcal{D} = \bigcup_s \mathcal{D}^s$), where scenario s's subset $\mathcal{D}^s = \{(x_i^s, x_i^a, y_i)\}_{i=1}^{|\mathcal{D}^s|}$ is obtained according to x_i^s. Here the whole feature set x_i is divided as scenario-specific feature set x_i^s and scenario-agnostic feature set x_i^a. The scenario-specific features in x_i^s can be context features such as business IDs or exhibition position IDs, and can also be extended to user profile features (e.g., gender, age group) or item features (e.g., category, brand), which may result in different behaviour or exposure distributions. Splitting each scenario subset $\mathcal{D}^s$ as a training set $\mathcal{D}^s_{\text{train}}$ and a testing set $\mathcal{D}^s_{\text{test}}$, we have $\mathcal{D}_{\text{train}} = \bigcup_s \mathcal{D}^s_{\text{train}}$ and $\mathcal{D}_{\text{test}} = \bigcup_s \mathcal{D}^s_{\text{test}}$. The goal of MS-CTR prediction is to construct a unified CTR model based on $\mathcal{D}_{\text{train}}$ that can give accurate CTR prediction for all scenario subsets in $\mathcal{D}_{\text{test}}$.

4 METHODOLOGY

4.1 Architecture Overview

In order to model the specialities of feature interactions across multiple scenarios, we propose SATrans for MS-CTR prediction. As shown in Figure 1, SATrans stacks self-attention based interacting layers as the backbone, and consists of two scenario-specific components: (1) the **Scenario Encoder**, which transforms scenario-specific features into a fixed-length embedding vector, (2) the **Scenario-Adaptive Interaction (SAI) Layers**, which perform high-order feature interaction via the scenario-adaptive self-attention mechanism. Given the input feature set $\{x_i^s, x_i^a\}$, we first transform it as a sparse feature vector:

$$\mathbf{x} = [\mathbf{x}^s; \mathbf{x}^a] = [\mathbf{x}_1^s; \dots; \mathbf{x}_M^s; \mathbf{x}_1^a; \dots; \mathbf{x}_{N-M}^a], \tag{1}$$

where M is the number of scenario-specific features, and N is the number of all features. After that, we first feed the scenario-specific features $\mathbf{x}^s$ into the scenario encoder to obtain the scenario embedding $\mathbf{s}$, then projects all features $\mathbf{x}$ into the same low-dimensional space with an embedding layer and obtain the dense embeddings $\mathbf{e} = [\mathbf{e}_1; \dots; \mathbf{e}_N]$, followed by multiple scenario-adaptive interacting layers, where high-order features are combined through the self-attention mechanism under the guidance of the scenario embedding. By stacking l interacting layers, scenario-adaptive feature interaction up to $(l+1)$ orders can be modeled. The output of the final interacting layer is concatenated, followed by a linear layer and a sigmoid function to estimate the CTR. The key points of SATrans are how to design the effective scenario encoder and the scenario-adaptive interaction module. In the following sections, we introduce the details of our proposed method.

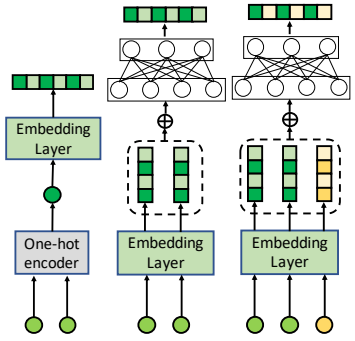


Figure 2: Three types of Scenario Encoder

4.2 Scenario Feature Encoder

Given the scenario-specific features $\mathbf{x}^s = [\mathbf{x}_1^s; \dots; \mathbf{x}_M^s]$, a scenario-adaptive encoder is utilized for encoding the scenario features as a fix-length scenario embedding $\mathbf{s} \in \mathbb{R}^L$, to guide the feature interaction in each SAI-layer, where the dimension L is dependent on the specific implementation of SAI-layers. In order to improve the quality of the scenario embedding, we consider three sources of information: 1) Scenario speciality information, which distinguishes different scenarios; 2) Shared knowledge, which encodes the commonalities among scenarios; 3) Structural position, which represents the position (e.g., the depth of the current layer, upon query or key embeddings) in which the scenario embedding is involved upon the self-attention network. We propose three types of implementations with regard to different sources of information.

Independent Embedding (IE): This approach first transforms $\mathbf{x}^s$, the concatenated sparse vectors of scenario features, into a one-hot sparse feature $\mathbf{x}^o$ via an injective mapping and then uses an embedding matrix to project it into a low-dimensional vector $\mathbf{s}$. This practice considers each possible combination of all scenario feature fields as a scenario and uses independent embedding to represent each scenario, which means there is no knowledge shared among scenarios. Worse still, the embedding matrix may be large when the combinational number of features increases, which would be parameter-inefficient and inflexible.

Encoding Network (EN): In order to encode scenario features more flexibly and involve shared knowledge, we consider leveraging a shared encoding network to transform the scenario features. For each scenario feature field, the sparse feature vector $\mathbf{x}_i^s$ is first projected into a low-dimensional vector $\mathbf{e}_i^s$ using an embedding matrix $\mathbf{E}_i^s$. We concatenate the embedding vectors of each field as $\mathbf{e}^s = [\mathbf{e}_1^s; \mathbf{e}_2^s; \dots; \mathbf{e}_M^s]$, followed by a non-linear activation layer (e.g., ReLU [1]). Then we feed the embedding vector into a shared encoding Network $f_e(\cdot)$ to obtain the final scenario embedding $\mathbf{s}$. In our experiments, we found a simple matrix transformation is sufficient, which means $\mathbf{s} = \mathbf{W}_s \text{ReLU}(\mathbf{e}^s)$.

Encoding Network with Structural Position IDs (ENP): Since the scenario embedding is operated on different interacting layers and different positions (e.g., query or key) in the backbone self-attention network, it is reasonable to generate position-aware scenario embeddings to improve the expressive power of the network. To this end, apart from scenario features, we also feed a position ID as an extra feature to generate a unique scenario embedding for each structural position in SAI-layers. Specifically, we have

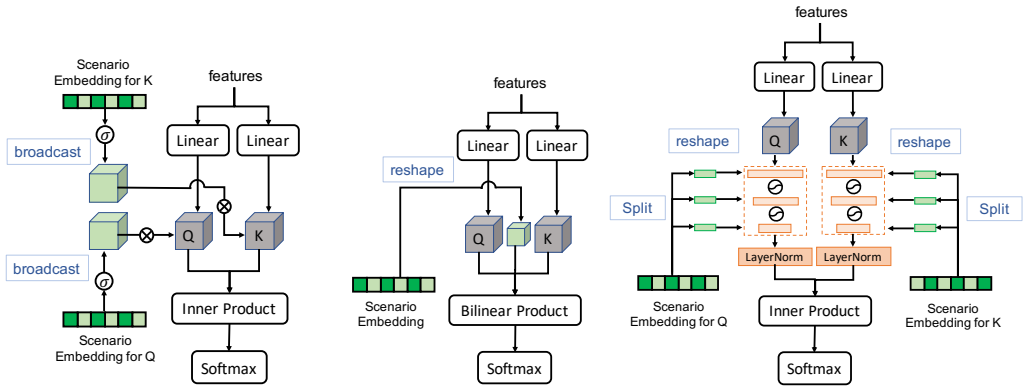
$$\mathbf{s}_{l,h} = \mathbf{W}_s \text{ReLU}(\text{Concat}(\mathbf{e}^s, \mathbf{p}_{l,h})), \tag{2}$$

where $\mathbf{p}_{l,h}$ is the position embedding, l is the layer depth ID, and $h \in \{Q, K\}$.

In terms of both **EN** and **ENP**, the individual network parameters for each scenario (or scenario feature) is just an embedding vector in low dimension (32 in our experiments), which is pretty parameter-efficient compared with MTL approaches where each scenario has individual gate networks, specific expert networks or output tower, making SATrans feasible for a large number of scenarios. We compare the parameter complexity in our experiments. In the following sections, we omit the subscripts l and use $\mathbf{s}_\text{Q}$ and $\mathbf{s}_\text{K}$ to denote the scenario embedding for Query and Key representation, respectively, for simplification. Note that $\mathbf{s}_\text{Q} = \mathbf{s}_\text{K}$ for **IE** and **EN** strategies.

4.3 Scenario-Adaptive Interacting Layer

Once we have the feature embeddings $\mathbf{e} = [\mathbf{e}_1; \dots; \mathbf{e}_N]$ in the same low-dimensional space, and the scenario embeddings $\mathbf{s}_{i,j}$ for each position in the interacting layers, we move to model scenario-adaptive high-order combinatorial features. Assuming the input representation of i-th feature is $\mathbf{h}_i$, and $\mathbf{h}_i = \mathbf{e}_i$ in the first interacting layer. We first introduce the multi-head self-attention mechanism to determine the importance of each feature combination. Taking



(a) SA-Gate **(b) SA-Bilinear** **(c) SA-MetaNet**

Figure 3: Three strategies of calculating scenario-adaptive self-attention

the i-th feature as an example, firstly, the correlation between the i-th feature and the j-th feature ($i, j \in \{1, \ldots, N\}$) under a specific attention head h is defined as

$$\alpha_{i,j}^{(h)} = \frac{\exp(\phi^{(h)}(\mathbf{h}_i, \mathbf{h}_j))}{\sum_k^N \exp(\phi^{(h)}(\mathbf{h}_i, \mathbf{h}_k)} \tag{3}$$

$$\phi^{(h)}(\mathbf{h}_i, \mathbf{h}_j) = \langle \mathbf{W}_{\mathrm{Q}}^{(h)} \mathbf{h}_i, \mathbf{W}_{\mathrm{K}}^{(h)} \mathbf{h}_j \rangle \tag{4}$$

where $\phi^{(h)}(\cdot, \cdot)$ is an attention function which defines the unnormalized correlation between i-th feature and j-th feature under head h. It can be a neural network or simply an inner product, i.e., $\langle \cdot, \cdot \rangle$. $\mathbf{W}_{\mathrm{Q}}^{(h)}, \mathbf{W}_{\mathrm{K}}^{(h)} \in \mathbb{R}^{d' \times d}$ are transformation matrices which project the original embedding space $\mathbb{R}^d$ into a new space $\mathbb{R}^{d'}$. $d' = d/H$ and H is the number of attention heads. Then, the representation of i-th feature in subspace h is updated via aggregating other features by coefficients $\alpha_{i,j}$:

$$\hat{\mathbf{h}}_i^{(h)} = \sum_l^M \alpha_{i,j}^{(h)} (\mathbf{W}_{\mathrm{V}}^{(h)} \mathbf{h}_l), \tag{5}$$

where $\mathbf{W}_{\mathrm{V}}^{(h)} \in \mathbb{R}^{d' \times d}$. Therefore, $\hat{\mathbf{h}}_i^{(h)}$ is a combination of i-th feature and its relevant features under head h. The correlation function in Equation 4 treats instances from all scenarios in the same way, which ignores the distribution discrepancy between different scenarios. In order to model the distribution shifts across scenarios, we involve the scenario embeddings in the calculation of correlation coefficients between features. We first partition the scenario embeddings $\mathbf{s}_{\mathrm{Q}}, \mathbf{s}_{\mathrm{K}}$ into H parts, i.e., $\mathbf{s}_{\mathrm{Q}} = [\mathbf{s}_{\mathrm{Q}}^{(1)}, \ldots, \mathbf{s}_{\mathrm{Q}}^{(H)}], \mathbf{s}_{\mathrm{K}} = [\mathbf{s}_{\mathrm{K}}^{(1)}, \ldots, \mathbf{s}_{\mathrm{K}}^{(H)}]$, where $\mathbf{s}_{\mathrm{K}}^{(h)}, \mathbf{s}_{\mathrm{Q}}^{(h)} \in \mathbb{R}^{L/H}$, then improve the scenario-adaptive attention function under head h as:

$$\phi_{sa}^{(h)}(\mathbf{h}_i, \mathbf{h}_j, \mathbf{s}_{\mathrm{Q}}^{(h)}, \mathbf{s}_{\mathrm{K}}^{(h)}). \tag{6}$$

Now, the problem is how to design the scenario-adaptive attention function $\phi_{sa}^{(h)}(\cdot, \cdot, \cdot, \cdot)$, which would impact the interaction quality significantly. Based on the order of computational complexity from lowest to highest, we considered three approaches as illustrated in Figure 3:

- **SA-Gate (Bit-wise):** A direct strategy to involve scenario embedding via bit-wise transformation is the gate mechanism. Specifically, we generate the Gate modules based on scenario embeddings to filter the feature embeddings:

$$\phi_{sa}^{(h)}(\mathbf{h}_i, \mathbf{h}_j, \mathbf{s}_{\mathrm{Q}}^{(h)}, \mathbf{s}_{\mathrm{K}}^{(h)}) = \langle \sigma(\mathbf{s}_{\mathrm{Q}}^{(h)}) \circ (\mathbf{W}_{\mathrm{Q}}^{(h)} \mathbf{h}_i), \sigma(\mathbf{s}_{\mathrm{K}}^{(h)}) \circ (\mathbf{W}_{\mathrm{K}}^{(h)} \mathbf{h}_j) \rangle \tag{7}$$

  where $\sigma(x) = 1/(1 + e^{(-x)})$ denotes Sigmoid Function, and $\circ$ denotes element-wise product.
- **SA-Bilinear (Bilinear):** This method performs a bilinear transformation upon feature embeddings, parameterized by a scenario-aware matrix $\mathbf{S}$ reshaped from the scenario embedding. The attention score is calculated as:

$$\phi_{sa}^{(h)}(\mathbf{h}_i, \mathbf{h}_j, \mathbf{s}_{\mathrm{Q}}^{(h)}, \mathbf{s}_{\mathrm{K}}^{(h)}) = (\mathbf{W}_{\mathrm{Q}}^{(h)} \mathbf{h}_j)^\top \mathbf{S} (\mathbf{W}_{\mathrm{K}}^{(h)} \mathbf{h}_i), \tag{8}$$

$$\text{where } \mathbf{S} = \mathrm{Reshape}(\mathbf{s}_{\mathrm{Q}}^{(h)}) \in \mathbb{R}^{d \times d}. \tag{9}$$

  In this strategy, $\mathbf{s}_{\mathrm{Q}}^{(h)}$ and $\mathbf{s}_{\mathrm{K}}^{(h)}$ are identical in each layer.
- **SA-MetaNet (Nonlinear):** The previous two strategies adopt bit-wise and bilinear transformation to involve scenario features, which have limited expressive capability and might fail to model the complicated relationships between scenario information and the interacting features. To this end, we consider non-linear transformation via a MetaNet mechanism, which is similar to dynamic weight units [30]. Take $\mathbf{s}_{\mathrm{Q}}^{(h)}$ as an example, it is first partitioned into P slots $[\mathbf{s}_{\mathrm{Q},1}^{(h)}; \ldots; \mathbf{s}_{\mathrm{Q},P}^{(h)}]$ to generate the projection parameters of a P-layer Meta Network $f_{\mathbf{s}_{\mathrm{Q}}^{(h)}}^{\mathrm{m}}(\cdot)$:

$$f_{\mathbf{s}_{\mathrm{Q}}^{(h)}}^{\mathrm{m}} = \mathbf{W}_1 \delta(\mathbf{W}_2 \delta(\ldots \delta(\mathbf{W}_P \mathbf{x}) \ldots))$$

$$\text{where } \mathbf{W}_p = \mathrm{Reshape}(\mathbf{s}_{\mathrm{Q},p}^{(h)}), \mathbf{W}_p \in \mathbb{R}^{d_{p-1} \times d_p},$$

  where d_p denotes the input dimension of $(p+1)$-th layer, δ is the non-linear activation function (i.e., ReLU). We construct $f_{\mathbf{s}_{\mathrm{K}}^{(h)}}^{\mathrm{m}}(\cdot)$ using $\mathbf{s}_{\mathrm{K}}^{(h)}$ with the same process. The generated MetaNets are used to transform the input feature embeddings before pair-wise interaction. Intuitively, different slots

of $\mathbf{s}_{\mathrm{Q}}$ and $\mathbf{s}_{\mathrm{K}}$ along with the activation function can be considered as scenario-aware filters from low-level to high-level upon the feature embedding, endowing the network with the capability to capture the implicit distinctions between scenarios. Now, the scenario-adaptive attention score is calculated as:

$$\phi_{sa}^{(h)}(\mathbf{h}_i, \mathbf{h}_j, \mathbf{s}_{\mathrm{Q}}^{(h)}, \mathbf{s}_{\mathrm{K}}^{(h)}) = \langle \mathrm{LN}_{\mathrm{Q}}^{(h)}(f_{\mathbf{s}_{\mathrm{Q}}^{(h)}}^{\mathrm{m}}(\mathbf{W}_{\mathrm{Q}}^{(h)}\mathbf{h}_i)), \mathrm{LN}_{\mathrm{K}}^{(h)}(f_{\mathbf{s}_{\mathrm{K}}^{(h)}}^{\mathrm{m}}(\mathbf{W}_{\mathrm{K}}^{(h)}\mathbf{h}_j))\rangle, \tag{10}$$

where $\mathrm{LN}_{\mathrm{Q}}^{(h)}(\cdot)$ and $\mathrm{LN}_{\mathrm{K}}^{(h)}(\cdot)$ are the layer normalization layers to normalize the embedding distribution, which have independent parameters across layers. We found the normalization layers are essential because after multiple layers of non-linear transformation, the variance of embeddings will be significantly magnified, which impairs the convergence drastically. In practice, we move the MetaNet along with LN layer before the partition of multiple heads, which allows the information interaction across different heads and empirically achieves better performance. Therefore, the attention score under attention head h is denoted as:

$$\phi_{sa}^{(h)}(\mathbf{h}_i, \mathbf{h}_j, \mathbf{s}_{\mathrm{Q}}, \mathbf{s}_{\mathrm{K}}) = \langle [\mathrm{LN}_{\mathrm{Q}}(f_{\mathbf{s}_{\mathrm{Q}}}^{\mathrm{m}}(\mathbf{W}_{\mathrm{Q}}\mathbf{h}_i)]^h, [\mathrm{LN}_{\mathrm{K}}(f_{\mathbf{s}_{\mathrm{K}}}^{\mathrm{m}}(\mathbf{W}_{\mathrm{K}}\mathbf{h}_j)]^h)\rangle, \tag{11}$$

where $\mathbf{W}_Q$ and $\mathbf{W}_K \in \mathbb{R}^{d\times d}$ are the transformation matrices, $[\cdot]^h$ denotes the partition operation and selection of h-th subspace.

Based on Equation 5, we update the representation of i-th feature under attention head h as $\hat{\mathbf{h}}_i^h$, and then aggregate the features from different subspaces as follows:

$$\hat{\mathbf{h}}_i = \hat{\mathbf{h}}_i^1 \oplus \hat{\mathbf{h}}_2^h \ldots \oplus \hat{\mathbf{h}}_H^h, \tag{12}$$

where $\oplus$ is the concatenation operator. Next, we use a projection matrix $\mathbf{W}_{\mathrm{Agg}}$ to transform the learned features and add standard residual connections to preserve previously learned combinatorial features including raw individual features (i.e., first-order), followed by a Layer Normalization layer. Formally, the output representation of i-th feature is

$$\mathbf{h}_i^{\mathrm{O}} = \mathrm{LN}(\mathbf{W}_{\mathrm{A}}\hat{\mathbf{h}}_i + \mathbf{h}_i). \tag{13}$$

With such an interacting layer, each feature representation will be updated into a new feature space, with information aggregated from other fields under the guidance of scenario information. We can stack multiple such layers to model arbitrary-order combinatorial features. We concatenate the output embedding of the last layer to obtain $\mathbf{h}^{\mathrm{Out}} = \mathbf{h}_1^{\mathrm{Out}} \oplus \mathbf{h}_2^{\mathrm{Out}} \ldots \oplus \mathbf{h}_N^{\mathrm{Out}}$ and use a linear layer with a Sigmoid function σ to obtain the final prediction:

$$p_{\mathrm{CTR}} = \sigma(\mathbf{W}_{\mathrm{O}}\mathbf{h}^{\mathrm{Out}} + \mathbf{b}_{\mathrm{O}}), \tag{14}$$

where $\mathbf{W}_{\mathrm{O}} \in \mathbb{R}^{1\times Nd}$ and $\mathbf{b}_{\mathrm{O}} \in \mathbb{R}$. The entire network is optimized by cross-entropy loss. The analysis of both space and time complexity is detailed in Appendix A.

5 EXPERIMENTS

In this section, we conduct experiments on both public and industrial datasets to evaluate the proposed SATrans and answer the following questions:

- **RQ1:** How does the SATrans perform compared with the baseline models?
- **RQ2:** How does each proposed module contribute to the performance?
- **RQ3:** Is the proposed model sensitive to selection of the scenario indicator?
- **RQ4:** Is the proposed model scalable to a large number of scenarios?
- **RQ5:** Does the proposed model capture the difference in feature interaction between scenarios? Is it explainable?

5.1 Experimental Setup

5.1.1 Datasets and Evaluation Protocals. We conduct experiments on both public datasets (Ali-CCP and Ali-Mama) and an industrial dataset named as WeChat-MS. The description and statistics of the datasets are detailed in B.1. We apply the most commonly-used **AUC** (Area Under the ROC)[3] to evaluate the model and illustrate the AUC score for each scenario individually. Especially, we also show the overall score over the whole dataset for Ali-CCP and Ali-Mama since they can be evaluated for general CTR prediction.

5.1.2 Baselines. To demonstrate the effectiveness of our proposed model, we compare SATrans with three classes of previous models. (A) Single-scenario Feature Interaction (FI) approaches, which include DCN [24], DeepFM [4], xDeepFM [10], NFM [5], AFM [27], AutoInt [21], and we evaluate their performance with three heuristic training strategies: 1) *Mix:* Train a shared model using samples from all scenarios; 2) *Separate:* Train a separate model for each scenario individually. 3) *Finetune:* Train a shared model using all samples and then finetune the model for each scenario using its own samples. (B) Multi-Task Learning models, which include Shared Bottom [17], MMOE [11], PLE [22]. (C) Multi-Scenario Modeling models, including STAR [20] and AdaSparse [29]. The implementation details of these methods are introduced in B.2

5.2 Overall Performance Comparison (RQ1)

We show the overall experimental results of Ali-CCP and Ali-Mama in Table 1, and Industrial dataset in Table 2. Note that we only show SATrans with MetaNet+EN or MetaNet+ENP here, which have the best performance among all combinations. We find the following observations:

(1) Singe-scenario FI approaches have diverse performances on the three datasets. For example, AFM achieves the best overall performance (0.6164) on Ali-CCP, AutoInt ranks 1st (0.6821) on Ali-Mama, and performs best in two scenarios on WeChat-MS. Different training strategies also show different patterns over the datasets. In general, Mix training achieves better performance than separate training and finetuning, but the opposite results are observed from WeChat-MS. We guess it is because Ali-CCP and Ali-Mama are naturally general CTR prediction datasets and the distributions of different scenarios are similar. In contrast, the scenarios in WeChat-MS are set by different businesses, which may follow quite distinctive patterns and require stronger speciality modeling. Finetuning the model for each scenario improves the performance compared with Mix training in some cases (e.g., Scenario 3 on Ali-CCP), but the AUC declines in general. It

Table 1: The overall performance over Ali-CCP and Ali-Mama datasets in terms of AUC. Boldface denotes the highest score and underline indicates the best result of the baselines. ⋆ represents significance level p-value < 0.05. The three values of each cell for FI models denote the AUC scores of Mix, Separate and Finetune strategies, respectively. ↑ denotes that the value is larger than that of the corresponding Mix training strategy, and ↓ otherwise.

Methods \ Datasets		Ali-CCP				Ali-Mama				
Group	Model	All	S1	S2	S3	All	S1	S2	S3	S4
FI (Mix& Separate& Finetune)	DCN	0.6097	0.6121(0.6124↑)(0.6114↓)	0.6081(0.6068↓)(0.6063↓)	0.5920(0.5755↓)(0.5906↓)	0.6279	0.6176(0.6043↓)(0.6132↓)	0.6129(0.5920↓)(0.6080↓)	0.6236(0.6102↓)(0.6176↓)	0.6301(0.6299↓)(0.6255↓)
	DeepFM	0.6083	0.6103(0.6105↑)(0.6107↑)	0.6071(0.6056↓)(0.6059↓)	0.5913(0.5731↓)(0.5904↓)	0.6254	0.6154(0.6019↓)(0.6104↓)	0.6116(0.5946↓)(0.6058↓)	0.6211(0.6092↓)(0.6153↓)	0.6274(0.6271↓)(0.6271↓)
	xDeepFM	0.6109	0.6140(0.6100↓)(0.6113↓)	0.6092(0.6052↓)(0.6057↓)	0.5914(0.5714↓)(0.5927↑)	0.6273	0.6169(0.6027↓)(0.6098↓)	0.6124(0.5949↓)(0.6012↓)	0.6228(0.6099↓)(0.6152↓)	0.6295(0.5950↓)(0.6302↑)
	NFM	0.6126	0.6128(0.6074↓)(0.6118↓)	0.6123(0.6086↓)(0.6101↓)	0.5923(0.5661↓)(0.5957↑)	0.6149	0.6077(0.5689↓)(0.6074↓)	0.6079(0.5625↓)(0.6080↑)	0.6140(0.5832↓)(0.6136↓)	0.6159(0.6171↑)(0.6150↓)
	AFM	0.6164	0.6165(0.6067↓)(0.6152↓)	0.6162(0.6128↓)(0.6146↓)	0.5936(0.5710↓)(0.5962↑)	0.6091	0.6045(0.4909↓)(0.6034↓)	0.6033(0.5031↓)(0.6030↓)	0.6084(0.5130↓)(0.6080↓)	0.6098(0.6079↓)(0.6041↓)
	AutoInt	0.6117	0.6143(0.6119↓)(0.6106↓)	0.6103(0.6045↓)(0.6066↓)	0.5943(0.5710↓)(0.5929↓)	0.6281	0.6189(0.6031↓)(0.6128↓)	0.6140(0.5945↓)(0.6071↓)	0.6239(0.6089↓)(0.6169↓)	0.6302(0.6290↓)(0.6262↓)
MTL&MSM	Shared Bottom	0.6162	0.6190	0.6150	0.5962	0.6268	0.6177	0.6133	0.6227	0.6290
	MMOE	0.6168	0.6218	0.6152	0.5971	0.6271	0.6156	0.6118	0.6224	0.6296
	PLE	0.6179	0.6213	0.6159	0.5988	0.6276	0.6173	0.6126	0.6218	0.6303
	STAR	0.6152	0.6189	0.6144	0.5911	0.6177	0.6145	0.6095	0.6181	0.6274
	AdaSparse	0.6169	0.6190	0.6151	0.5992	0.6273	0.6180	0.6141	0.6233	0.6293
SATrans	MetaNet+EN	**0.6228**⋆	**0.6239**⋆	**0.6223**⋆	**0.5998**⋆	0.6313⋆	0.6221⋆	0.6178⋆	0.6266⋆	0.6334⋆
	MetaNet+ENP	0.6201⋆	0.6216⋆	0.6192⋆	0.5966⋆	**0.6337**⋆	**0.6254**⋆	**0.6212**⋆	**0.6291**⋆	**0.6355**⋆

Table 2: The overall performance over WeChat-MS.

Group	Model	S1	S2	S3
FI (Mix& Separate& Finetune)	DCN	0.7908(0.7984↑)(0.7815↓)	0.7461(0.7567↑)(0.7486↑)	0.7124(0.7155↑)(0.7142↑)
	DeepFM	0.7955(0.7967↑)(0.7958↑)	0.7462(0.7569↑)(0.7540↑)	0.7209(0.7163↓)(0.7142↓)
	xDeepFM	0.7972(0.7992↑)(0.7956↓)	0.7477(0.7569↑)(0.7562↑)	0.7118(0.7167↑)(0.7127↑)
	NFM	0.7929(0.7965↑)(0.7943↑)	0.7376(0.7464↑)(0.7463↑)	0.7008(0.7056↑)(0.7055↑)
	AFM	0.7803(0.7838↑)(0.7778↓)	0.7301(0.7152↓)(0.7391↓)	0.6923(0.6940↑)(0.6930↓)
	AutoInt	0.8013(0.8003↓)(0.7784↓)	0.7502(0.7558↑)(0.7366↓)	0.7151(0.7173↑)(0.6971↓)
MTL&MSM	Shared Bottom	0.7975	0.7304	0.7107
	MMOE	0.7992	0.7347	0.7141
	PLE	0.8022	0.7569	0.7182
	STAR	0.7984	0.7525	0.7159
	AdaSparse	0.7971	0.7535	0.7147
SATrans	MetaNet+EN	**0.8082**⋆	0.7627⋆	**0.7214**⋆
	MetaNet+ENP	0.8066⋆	**0.7642**⋆	0.7197⋆

may be attributed to the catastrophic forgetting issue, where the model tends to overfit the specialities of a specific scenario while forgetting the common knowledge learned from Mix training. The limitations of FI models with heuristic training strategies motivate joint modeling of scenario distinctions and commonalities.

(2) The MTL and MSM models are generally superior to singe-scenario FI approaches on Ali-CCP, as we can see that all MTL&MSM models have an overall AUC higher than 0.6150, while for single-scenario FI models, only the AUC of AFM is above 0.6150. However, the difference on Ali-Mama is relatively marginal, and the best AUC among baselines is achieved by AutoInt, which is 0.6281. PLE achieves the best performance in two scenarios on WeChat-MS, but the overall superiority of MTL&MSM methods is not obvious as well. These observations illustrate that the advantages of feature interaction modeling and scenario speciality modeling vary with the settings and datasets, and imply the necessity to study scenario-aware feature interaction approaches which enjoy the merits of both.

(3) SATrans with both MetaNet-EN and MetaNet-ENP settings consistently outperform all the baselines by a large margin: The best setting of SATrans improves the best baseline by 0.80% and 0.89% in terms of overall AUC on Ali-CCP and Ali-Mama, respectively, and 0.59% on average on WeChat-MS, which are a significant improvement in CTR prediction. Compared with FI models, SATrans employs the mixed data in a more elegant way which captures the scenario specialities with scenario-aware feature correlation scores while learning the commonalities with shared projection matrices and feature embeddings. Compared with MTL models and other MSM models, SATrans explicitly learns scenario-adaptive high-order feature interactions, which are beneficial to the prediction of CTR.

Table 3: Results of Ablation study. Table (a) compares different implementation strategies of Scenario Encoder (SE) and SAI-layers over Ali-CCP and Ali-Mama datasets. G denotes SA-Gate, B denotes SA-Bilinear, and M denotes SA-MetaNet. Table (b) shows the comparison of inserting MetaNet after different positions.

Modules		Dataset	
SAI	SE	Ali-CCP	Ali-Mama
\	\	0.6116	0.6281
G	IE	0.6119	0.6310
G	EP	0.6124	0.6311
G	ENP	0.6120	0.6302
B	IE	0.6121	0.6299
B	EP	0.6117	0.6309
B	ENP	0.6128	0.6303
M	IE	0.6169	0.6316
M	EP	**0.6228**	0.6313
M	ENP	0.6201	**0.6337**

(a)

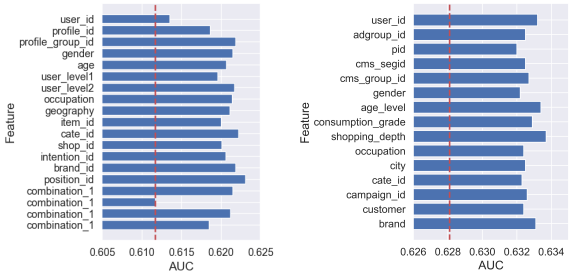
Position			Dataset	
Q	K	V	Ali-CCP	Ali-Mama
×	×	×	0.6116	0.6281
×	×	✓	0.6127	0.6239
×	✓	×	0.6162	0.6298
×	✓	✓	0.6078	0.6146
✓	×	×	0.6206	0.6325
✓	×	✓	0.6136	0.6248
✓	✓	×	**0.6228**	**0.6337**
✓	✓	✓	0.6102	0.6195

(b)

5.3 Ablation Study (RQ2)

To verify the effectiveness of each module in SATrans, we conduct a series of ablation studies over the public datasets in terms of overall AUC. We take stacked self-attention layers as the baseline of our proposed model.

5.3.1 Comparison of strategies for various modules. Since we propose three types of implementation strategies for both the Scenario Encoder and the SAI-Layer in Section 4.2 and Section 4.3, respectively, we compare their contributions to the prediction by evaluating all the combinations. As illustrated in Table 3(a), all the combinations outperform the vanilla self-attention interaction (baseline), indicating the effectiveness of involving the scenario information in the calculation of feature correlations. Generally, SA-MetaNet is superior to SA-Gate and SA-Bilinear, which shows that nonlinear transformation is more suitable to capture the feature correlations under the guidance of scenario features, indicating the complexity and challenge of scenario-adaptive feature interaction. In terms of scenario encoder, which can find that EN and ENP have better results than IE, which implies the importance of the shared encoding network, which encodes shared knowledge in the embeddings from different scenarios. EN achieves the best performance on Ali-CCP while ENP performs better in Alimama, and improves



(a) Ali-CCP (b) Ali-Mama

Figure 4: Comparison of selecting different features as the scenario indicator to partition the dataset. The red line denotes the results of AutoInt, and the blue bars denote the results of SATrans with different scenario indicators.

S2 in WeChat-MS, which implies the effects of position-aware embeddings vary with different datasets.

5.3.2 Insert Scenario-Adaptive modules at different positions. The architecture of self-attention layers is much more complicated than that of feed-forward neural networks, thus it is non-trivial to effectively inject the scenario information into the architecture. Table 3(b) compares the AUC performance of inserting MetaNet after different positions in the self-attention layers. We found the following observations: (1) inserting MetaNet after Queries always boosts the performance; (2) the contribution of inserting MetaNet after Keys is unstable; (3) inserting MetaNet after Values always harms the performance when combined with other strategies; (4) inserting MetaNet after both Queries and Keys have the best performance on both datasets. We speculate that the correlation calculation between features is more suitable to capture distinctions among scenarios, while feature projection parameters are prone to encode common knowledge.

5.4 Selection of the Scenario Indicator (RQ3)

In practice, the log data might not be naturally divided into different scenarios, or we can only access data from a single business. In this case, a common practice is to use some user/item group indicators such as gender, or age groups to partition the data into various scenarios. Therefore, the selection of the scenario indicator is important and might influence the overall performance significantly. As illustrated in Figure 4, we set different features as the scenario indicator to partition the dataset and compare the overall performance on both Ali-CCP and Ali-Mama. We can find that the proposed SATrans is robust to the selection of scenario features and can boost the performance by a large margin compared with vanilla self-attention for most features, which demonstrates the robustness and effectiveness of SATrans in single-scenario settings.

5.5 Parameter Complexity (RQ4)

In this section, we compare the models in terms of parameter complexity on Ali-Mama dataset. In order to evaluate models' scalability over the number of scenarios, we select five features: *pid* (2), *cms_group_id* (14), *cms_segid* (98), *cate_id* (6769) and *user_id* (1141729) as the scenario indicator, where the number in the brackets is the feature size (i.e., the number of scenarios). As we can see

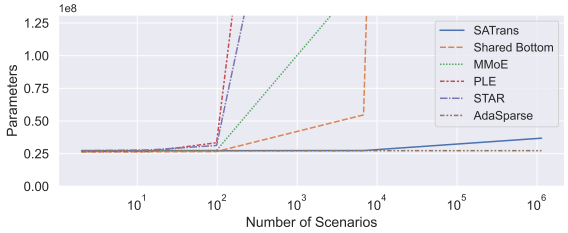


Figure 5: Space complexity with varying scenario numbers on the Ali-Mama dataset.

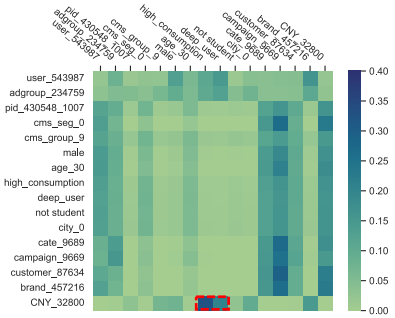
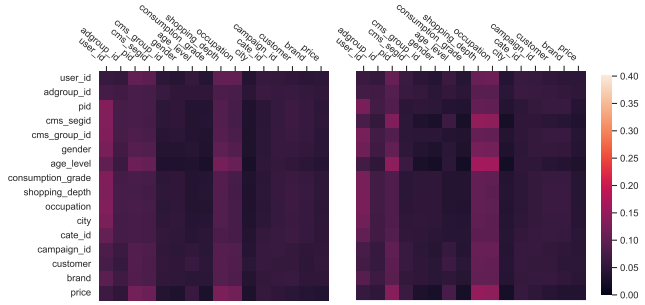


Figure 6: Label=1, Predicted CTR=0.19 (average CTR 0.05 in Ali-Mama)

in Figure 5, The parameters of Shared Bottom, MMoE and PLE grow rapidly with the increase of scenarios, and they are untrainable for *cate_id* and *user_id*. In contrast, the memory cost of SATrans increases marginally compared with others. It is because the only extra parameters are introduced by the scenario-specific feature embeddings, which are much smaller than embeddings of the entire feature set. The memory cost of AdaSparse is independent of the scenario number, but its prediction performance is limited due to the lack of scenario-adaptive feature interaction.



(a) Low-Consumption Scenario (b) High-Consumption Scenario

Figure 7: Average attention scores for low consumption scenario and high consumption scenario on Ali-Mama

5.6 Visualization and Interpretability (RQ5)

In this section, we visualize the attention scores of SATrans to present how our model provide both instance-level and scenario-level interpretability. Since most feature fields and values of Ali-CCP are anonymized, we only conduct fine-grained analysis on the interpretability of SATrans on Ali-Mama dataset, which has more detailed feature descriptions. Specifically, we study the interpretability from both instance level and scenario level as follows:

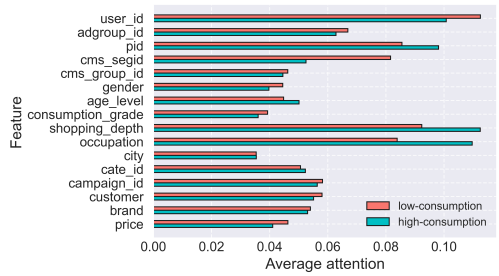


Figure 8: Average accumulated attention for low-consumption-grade and high-consumption-grade scenarios.

- **Instance-level interpretability:** Since SATrans adopt the self-attention mechanism as the interacting layers, it provides the same instance-level interpretability as AutoInt [21]. Figure 6 presents the correlations between different feature fields for a selected instance obtained by the attention score in the first interacting layer. We can see that SATrans is able to identify the meaningful combinatorial feature *<price=CNY_32800, consumption_grade*[4]*=high, shopping_depth*[5]*=deep>* (i.e., red dotted rectangle), which is reasonable since active high-consumption-grade users are more likely to buy expensive products.
- **Scenario-level interpretability:** To understand the distinction across different scenarios, we use *consumption_grade* as the scenario indicator and compare the average attention weights for low-consumption-grade and high-consumption-grade scenarios. As illustrated in Figure 7, in high-consumption-grade scenario, the model would focus more on some combinatorial features such as *<age_level, shopping_depth>*, *<age_level, occupation>* or *<cms_segid, occupation>*. We also average the attention weights by the vertical axis to show the importance of each feature in both scenarios. From Figure 8, we can conclude that in high-consumption-grade scenario, users' shopping depth (frequency) and occupation are important factors for prediction, which suggests that more fine-grained strategies can be applied to specific groups of people in re-ranking steps for an extra bonus. As far as we know, none of the existing MS-CTR approaches can provide such insights.

We also compare the attention patterns of SATrans and AutoInt and find that SATrans can generate more contrasting and meaningful attention scores compared with AutoInt. We detail this part in Appendix C.2 due to the limit of pages.

6 COMMERCIAL APPLICATION

To evaluate the performance of SATrans in real product environment, we conduct online A/B test on WeChat Official Account Recommendation platform for two consecutive weeks.

System Overview: As illustrated in Figure 13, the platform is composed of two core components: an offline training module and a Real-Time inference module. The log data from multiple scenarios are merged to periodically train a unified model. When a user accesses any scenarios, a request with the user's attributes and contextual features is sent to the Real-Time inference system. In the system, the recall module first retrieves hundreds of candidates, and then the ranking module leverages the features extracted from the database and the trained model to predict the scores. Finally, the candidate contents are sorted according to a pre-defined ranking function (e.g., CTR) and the top-k contents will be inserted into certain pre-determined positions for presentation. Note that the multi-scenario modeling is only utilized in the ranking stage, and each scenario has its own recall model. We elaborate on the online implementation details of SATrans in Appendix D.

[4]consumption grade of users, which can be low, middle, high and unknown

[5]shopping depth of users, which can be shallow, moderate, deep and unknown

Table 4: Online gains in three scenarios. Note that in the WeChat Official Account recommendation platform, 0.5% increase in CTR is a significant improvement.

	Average	Subscribed	Featured	Trending
CTR	2.84%	0.94%	4.76%	2.83%

Scenario Description: We deploy SATrans on three scenarios in WeChat Official Account Recommendation Platform, where billions of users are involved. Specifically, the three scenarios are Subscribed Feeds, which only pushes the contents from subscribed official accounts of users, Featured Feeds, which pushes contents based on certain personalized factors and Trending Feeds, which pushes popular contents in the candidate pool. The recommended contents are all presented in different spots on the subscription page in WeChat.

Results of Online A/B Testing: We conduct online A/B testing in the recommender system of WeChat from 2022-12 to 2023-01 with two week's testing. The compared baseline is a highly-optimized multi-scenario model. We use Click Through Rate (CTR) to evaluate the online effectiveness. As we can see in Table 4, SATrans contributes 2.84% CTR promotion on average compared to the baseline, which is a significant improvement in WeChat Official Account platform. The improvement over *Subscribed Feed* (0.94%) is relatively smaller than the other two (4.76% and 2.83%, respectively). We guess it is because the distribution of user behaviours in this scenario is relatively different from the other two, as the candidate pool of each user in this scenario is quite limited and only contains the contents pushed by subscribed official accounts of the user. This scenario has more regular user behaviours and benefits marginally from multi-scenario modeling, while the other two can benefit a lot from the signals of regular interests in it.

7 CONCLUSION

In this paper, we first summarize the limitations of existing models for MS-CTR prediction. Then, we propose scenario-adaptive feature interaction, which explicitly captures the distinction in feature interaction between scenarios via scenario-adaptive self-attention mechanism. SATrans consists of two modules: (1) Scenario Encoder, which encodes the scenario indicator or features into a fixed-length embedding; (2) Scenario-Adaptive Interacting Layers, which explicitly model the high-order interaction among features under the guidance of scenario embeddings. The proposed model possesses many merits: (1) Joint commonality modeling and distinction modeling, which contributes to a high prediction performance; (2) High Scalability, and it can be efficiently used when there are thousands of scenarios; (3) Both instance-level and scenario-level interpretability. SATrans has also been deployed in WeChat Official Account platform and serves billions of users.

REFERENCES

[1] Abien Fred Agarap. 2018. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375* (2018).

[2] Yuting Chen, Yanshi Wang, Yabo Ni, An-Xiang Zeng, and Lanfen Lin. 2020. Scenario-aware and Mutual-based approach for Multi-scenario Recommendation in E-Commerce. In *2020 International Conference on Data Mining Workshops (ICDMW)*. IEEE, 127–135.

[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] 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).

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

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

[7] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences* 114, 13 (2017), 3521–3526.

[8] Pengcheng Li, Runze Li, Qing Da, An-Xiang Zeng, and Lijun Zhang. 2020. Improving multi-scenario learning to rank in e-commerce by exploiting task relationships in the label space. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2605–2612.

[9] Zeyu Li, Wei Cheng, Yang Chen, Haifeng Chen, and Wei Wang. 2020. Interpretable click-through rate prediction through hierarchical attention. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 313–321.

[10] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 1754–1763.

[11] 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.

[12] Xiao Ma, Liqin Zhao, Guan Huang, Zhi Wang, Zelin Hu, Xiaoqiang Zhu, and Kun Gai. 2018. Entire space multi-task model: An effective approach for estimating post-click conversion rate. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 1137–1140.

[13] 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.

[14] 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.

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

[16] Steffen Rendle, Walid Krichene, Li Zhang, and John Anderson. 2020. Neural collaborative filtering vs. matrix factorization revisited. In *Fourteenth ACM conference on recommender systems*. 240–248.

[17] Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098* (2017).

[18] Ying Shan, T Ryan Hoens, Jian Jiao, Haijing Wang, Dong Yu, and JC Mao. 2016. Deep crossing: Web-scale modeling without manually crafted combinatorial features. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 255–262.

[19] Qijie Shen, Wanjie Tao, Jing Zhang, Hong Wen, Zulong Chen, and Quan Lu. 2021. SAR-Net: A Scenario-Aware Ranking Network for Personalized Fair Recommendation in Hundreds of Travel Scenarios. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 4094–4103.

[20] Xiang-Rong Sheng, Liqin Zhao, Guorui Zhou, Xinyao Ding, Binding Dai, Qiang Luo, Siran Yang, Jingshan Lv, Chi Zhang, Hongbo Deng, et al. 2021. One model to serve all: Star topology adaptive recommender for multi-domain ctr prediction. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 4104–4113.

[21] 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.

[22] Hongyan Tang, Junning Liu, Ming Zhao, and Xudong Gong. 2020. Progressive layered extraction (ple): A novel multi-task learning (mtl) model for personalized recommendations. In *Fourteenth ACM Conference on Recommender Systems*. 269–278.

[23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[24] 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.

[25] Ruoxi Wang, Rakesh Shivanna, Derek Z Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed H Chi. 2020. Dcn-m: Improved deep & cross network for feature cross learning in web-scale learning to rank systems. *arXiv preprint arXiv:2008.13535* (2020).

[26] Yichao Wang, Huifeng Guo, Bo Chen, Weiwen Liu, Zhirong Liu, Qi Zhang, Zhicheng He, Hongkun Zheng, Weiwei Yao, Muyu Zhang, et al. 2022. CausalInt: Causal Inspired Intervention for Multi-Scenario Recommendation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4090–4099.

[27] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. 2017. Attentional factorization machines: Learning the weight of feature interactions via attention networks. *arXiv preprint arXiv:1708.04617* (2017).

[28] Bencheng Yan, Pengjie Wang, Kai Zhang, Feng Li, Jian Xu, and Bo Zheng. 2022. APG: Adaptive Parameter Generation Network for Click-Through Rate Prediction. *arXiv preprint arXiv:2203.16218* (2022).

[29] Xuanhua Yang, Xiaoyu Peng, Penghui Wei, Shaoguo Liu, Liang Wang, and Bo Zheng. 2022. AdaSparse: Learning Adaptively Sparse Structures for Multi-Domain Click-Through Rate Prediction. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 4635–4639.

[30] Qianqian Zhang, Xinru Liao, Quan Liu, Jian Xu, and Bo Zheng. 2022. Leaving No One Behind: A Multi-Scenario Multi-Task Meta Learning Approach for Advertiser Modeling. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. 1368–1376.

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

[32] Yuanliang Zhang, Xiaofeng Wang, Jinxin Hu, Ke Gao, Chenyi Lei, and Fei Fang. 2022. Scenario-Adaptive and Self-Supervised Model for Multi-Scenario Personalized Recommendation. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 3674–3683.

A ANALYSIS OF SATRANS

A.1 Time Complexity

The time complexity of the scenario encoder is $O(d_s d^2 L_m)$. Within each interacting layer, the computation cost is three-fold. First, transforming the features using the MetaNet takes $O(NL_m d^2)$. Afterwards, calculating attention weights and forming combinatorial features under one head both cost $O(Ndd' + N^2 d')$ time respectively, and all the H heads takes $O(NHd'(N+d))$. The aggregation projection costs $O(Nd^2)$. Since $d = Hd'$ in our experiments, so the total computational cost of SATrans with L layers is $O(d_s d^2 L_m + (NL_m d^2 + N^2 d + Nd^2) \times L)$.

A.2 Space Complexity

The input embedding layer of the SAI-Layers contains Nd parameters, where N is the dimension of sparse representation of the input feature and d is the embedding dimension. Setting the hidden dimension of SAI-layers as d, the space complexity of projection matrices $\{\mathbf{W}_{\text{Query}}, \mathbf{W}_{\text{Key}}, \mathbf{W}_{\text{Value}}, \mathbf{W}_{\text{Agg}}\}$ is $O(Ld^2)$, where L denotes the number of layers. The output linear layer has Nd parameters. Since the complexity of *Scenario Encoders* is dependent on the implementation of *SAI-Layers*, we only analyze SA-MetaNet because it has the largest complexity. Assuming that the hidden dimension of the MetaNet is in the same scale of d and the depth is L_m, the dimension of the scenario embedding is $O(L_m d^2)$. Therefore, the space complexity of *Scenario Encoders* is $O(N_s^c L_m d^2)$ for IE strategy, where N_s^c is the dimension of the combined scenario feature. In terms of EN and ENP strategy, the space complexity of the encoding network is $O(L_m d^2 d_s)$, and the parameter number of the embedding layer of EN and ENP are $N_s d_s$ and $(N_s + N_p)d_s$, respectively, where d_s is the scenario embedding size, N_s is the dimension of sparse representation of scenario features and N_p is the number of positions. As far as the interacting layers and scenario encoding network are concerned, the space complexity is $O(Ld^2 + L_m d^2 d_s)$. Note that L, L_m, d, d_s are small (e.g., $L = 3, L_m = 2, d = d_s = 32$ in our experiments), thus SATrans is memory-efficient.

B EXPERIMENTAL SETTINGS

B.1 Datasets

- **Ali-CCP**[6][12]. The dataset is a public dataset released by Taobao with prepared training and testing set. We split the dataset into 3 scenarios according to the position_id as previous work [8].
- **Ali-Mama**[7]. This dataset is a public dataset released by Alimama, an online advertising platform in China. The dataset consists of 8 days of ad display/click logs from 2017-05-06 to 2017-05-12. We use the first 7 days for training and the last day for testing, We filter the samples of which user id is missing, and then divide the scenario according to *shopping_depth_id* as we found this scenario indicator leads to best performance for most approaches.
- **WeChat-MS**. The industrial dataset is sampled from the click logs of 3 scenarios (S1: *Subscribed Feeds*, S2: *Featured Feeds*, S3: *Trending Feeds*) in online recommendation platform of WeChat Official Account Service. Different scenarios have overlapped users and items. There are over 300 feature fields including user profiles, item features, context information as well as scenario-specific features. The training data and testing data are sampled from two consecutive days respectively.

The statistics of the three datasets are summarized in Table 5.

B.2 Implementation Details of Offline Experiments

All methods are implemented in pytorch[8] and deepctr_torch[9]. We empirically set the batch size as 8192. We use Adam [6] as the optimizer for all models and tune the learning rate from 1e-3 to 5e-3. The embedding size is tuned from [8,16,32,64,128], the dropout rate is searched from [0.1,0.2,. . .,0.5], the number of interacting layers is tuned from [2,3,4,5], the number of attention head is tuned from [2,4,8]. The hidden layers of other baselines are tuned from the combinations of [256,128,64], the number of experts are tuned from [3,4,5,10], and we set other hyper-parameters as default in deepctr_torch library.

Table 5: The percentage of instances, the average CTR, the number of training and testing instances of each scenario in Ali-CCP, Ali-Mama and WeChat-MS (We do not provided the CTR of WeChat-MS due to the policy of Tencent).

Dataset	Ali-CCP			
Scenario	All	S1	S2	S3
Percentage	100%	37.78%	61.46%	0.75%
CTR	3.89%	4.00%	3.81%	4.38%
#Train	42.30M	15.88M	26.09M	0.32M
#Test	43.02M	16.35M	26.34M	0.32M

Dataset	Ali-Mama				
Scenario	All	S1	S2	S3	S4
Percentage	100%	5.76%	4.21%	9.77%	80.26%
CTR	5.14%	5.33%	5.40%	5.17%	5.11%
#Train	20.01M	1.13M	0.83M	1.95M	16.10M
#Test	6.54M	0.40M	0.28M	0.64M	5.21M

Dataset	WeChat-MS			
Scenario	All	S1	S2	S3
Percentage	100%	47.96%	4.12%	44.11%
CTR	/	/	/	/
#Train	45.05M	21.89M	1.88M	19.11M
#Test	10.48M	4.74M	0.35M	5.30M

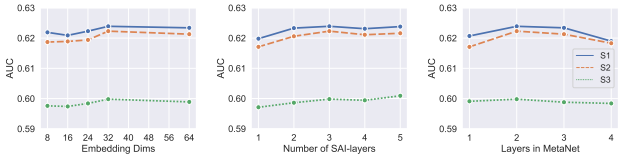


Figure 9: Hyper-parameter analysis on Ali-CCP

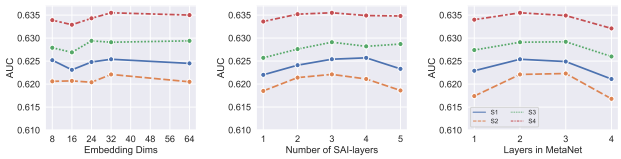
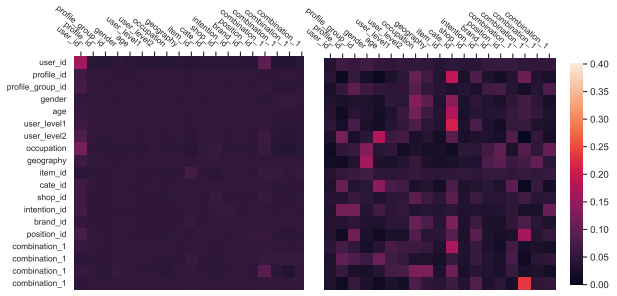


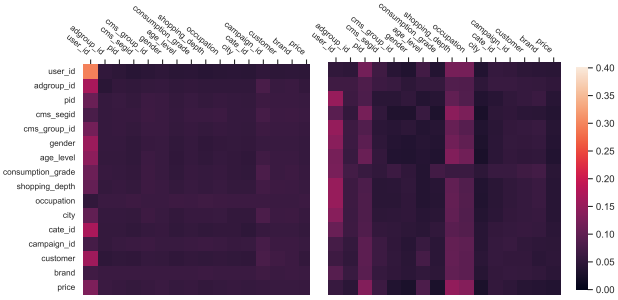
Figure 10: Hyper-parameter analysis on Ali-Mama

[6]https://tianchi.aliyun.com/dataset/dataDetail?dataId=408
[7]https://tianchi.aliyun.com/dataset/dataDetail?dataId=56
[8]https://pytorch.org/
[9]https://github.com/shenweichen/DeepCTR-Torch



(a) AutoInt (b) SATrans

Figure 11: Average attention scores on Ali-CCP



(a) AutoInt (b) SATrans

Figure 12: Average attention scores on Ali-Mama

C EXPERIMENTAL RESULTS

C.1 Sensitive Analysis of Hyper-parameters

For each dataset, we compare the performance of SATrans (MetaNet+EN for Ali-CCP and MetaNet+ENP for Ali-Mama) with different input embedding sizes, the effect of the number of SAI-layers and the depth of MetaNet. As we can see in Figure 9 and 10, SATrans is robust under embeddings of different dimensions, and even small dimension with only 8 still keeps an excellent performance. The performance of SATrans becomes better with the dimension increase, reaching the top at 3 layers and then declines as a too deep network leads to overfitting. The number of layers in the MetaNet module performs best when the value is 2. Compared with 1 layer, a 2-layer MetaNet provides more powerful non-linear transformation over feature interaction. However, performance drops when the number increases because it makes the overall network too deep and increases the difficulty of training and risk of overfitting.

C.2 Visualization of Average Attention Scores

In this section, we measure the correlations between the feature fields according to their average attention scores of the first interacting layer in the entire dataset for Ali-CCP and Ali-Mama, and compare our SATrans with AutoInt in Figure 11 and 12. We found the patterns of average attention scores of AutoInt and SATrans are quite different: 1) SATrans can generate more contrasting correlation scores between the feature fields, while AutoInt tends to learn even attention values and only one feature (i.e., *user_id*) is highlighted. This observation illustrates that SATrans are more good at distinguish the interactions among different features. 2) SATrans can capture crucial and meaningful feature interactions. For example, in Ali-CCP, we can see that *<user_consumption_grade, age>*, *<user_consumption_grade, category_id>* and *<profile_id, category_id>* are strongly correlated, which are meaningful feature combinatorial features. Besides, *<price, shopping_depth>*, *<price, occupation>* have high correlation scores, which are consistent with realistic observations.

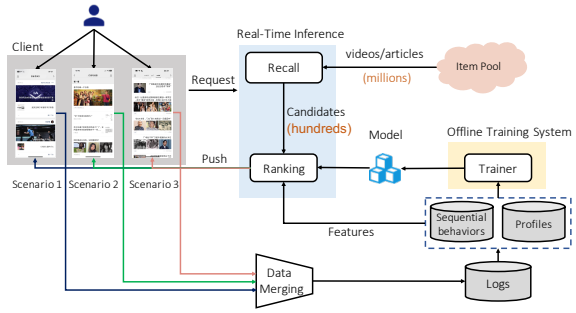


Figure 13: Overview of the Recommender System of WeChat Official Account Platform.

D IMPLEMENTATION DETAILS OF THE ONLINE MODEL

Figure 14 depicts the architecture of the entire model deployed in our online system. The input of the model is composed of multiple groups of features, which are encoded into fixed-length embeddings via different modules. The continuous numerical features are first normalized and discretized into multiple buckets, followed by an embedding layer. The historical sequence of a specific behaviour type is first represented by sequential modeling models (e.g., Transformer, LSTM) and then compressed into a fixed-length embedding using a pooling layer. Other categorical features are directly projected into dense vectors with an embedding layer. In addition to the scenario indicator, we also leverage 40+ indicators of user activities of the corresponding scenario as scenario features, which bring consistent improvements to the model performance (We also use these features as regular features in other baselines for fairness of comparison). The model is trained by 20 core Intel(R) Xeon(R) Platinum 8255C CPU @ 2.50GHz, 124GB RAM and 1 NVIDIA A100 Tensor Core GPU. All online models are trained with the same dataset, which is a combination of real-time user logs over multiple scenarios, and deployed on the same type of cluster servers.

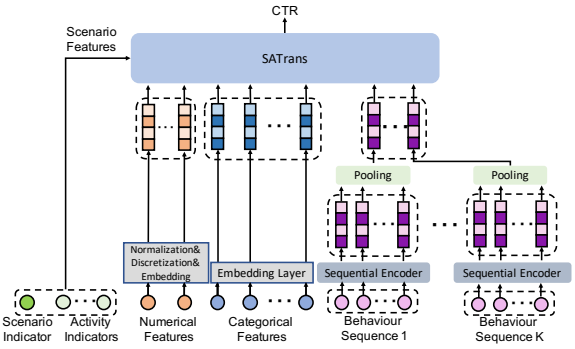


Figure 14: The overall model architecture in online system.