Disentangled Self-Attentive Neural Networks for Click-Through Rate Prediction

Yichen Xu\textsuperscript{1,*}, Yanqiao Zhu\textsuperscript{2,3,*}, Feng Yu\textsuperscript{4}, Qiang Liu\textsuperscript{2,3}, and Shu Wu\textsuperscript{2,3,5,†}

\textsuperscript{1}School of Computer Science, Beijing University of Posts and Telecommunications
\textsuperscript{2}Center for Research on Intelligent Perception and Computing, Institute of Automation, Chinese Academy of Sciences
\textsuperscript{3}School of Artificial Intelligence, University of Chinese Academy of Sciences
\textsuperscript{4}Alibaba Group
\textsuperscript{5}Artificial Intelligence Research, Chinese Academy of Sciences

linyxus@bupt.edu.cn, yanqiao.zhu@cripac.ia.ac.cn, yf271406@alibaba-inc.com, {qiang.liu, shu.wu}@nlpr.ia.ac.cn

ABSTRACT
Click-Through Rate (CTR) prediction, whose aim is to predict the probability of whether a user will click on an item, is an essential task for many online applications. Due to the nature of data sparsity and high dimensionality of CTR prediction, a key to making effective prediction is to model high-order feature interaction. An efficient way to do this is to perform inner product of feature embeddings with self-attentive neural networks. To better model complex feature interaction, in this paper we propose a novel Disentangled Self-Attentive Neural Networks (DESTINE) framework for CTR prediction that explicitly decouples the computation of unary feature importance from pairwise interaction. Specifically, the unary term models the general importance of one feature on all other features, whereas the pairwise interaction term contributes to learning the pure impact for each feature pair. We conduct extensive experiments using two real-world benchmark datasets. The results show that DESTINE not only maintains computational efficiency but achieves consistent improvements over state-of-the-art baselines.

KEYWORDS
Click-through rate prediction; high-order feature interaction; disentangled self-attention

1 INTRODUCTION
Click-Through Rate (CTR) prediction, seeking to predict the probability that a user will interact with a candidate item, is essential for many online applications, such as computational advertising and/or a fee. Request permissions from permissions@acm.org.

Figure 1: Our proposed disentangled self-attentive networks for CTR prediction that decouple the learning of the pairwise term and the unary term.

[11] and recommender systems [2]. One major challenge of making accurate prediction is that the data used in CTR tasks usually involve numerous categorical features, e.g., categories of ads, users’ device models, etc., and thus are high-dimensional and extremely sparse, distinct from continuous numerical features such as images. With such high-dimensional and sparse features as input, one complex model would be inevitably prone to overfitting. Therefore, a successful solution to extracting useful information from these high-dimensional data is to model combinatorial interaction among feature fields based on embedding lookup techniques, also known as \textit{cross features}. For example, for movie CTR prediction, one informative feature based on third-order feature interaction could be \{Age, Gender, Genre\}, considering that young men tend to prefer action movies. However, it is not possible to enumerate all combinatorial feature interaction due to the exponential complexity [2, 14, 20]. How to automatically model high-order feature interaction thereby attracts a lot of interests.

Recent development in CTR prediction has witnessed a transition from simple linear models [13] to more sophisticated methods that model arbitrary-order interaction among these sparse categorical features to make effective prediction. To model implicit feature interaction, the pioneering work Factorization Machines (FM) [12] proposes to model second-order feature interaction via inner-product of embedding vectors. Following this line, many other methods extend second-order FM to model higher-order interaction [1, 4, 10]. However, these methods suffer from high computational complexity, limiting its practical application in real world.

One widely-used solution to modeling high-order feature interaction is to compute inner product of feature embeddings [7, 16], which resembles self-attentive neural networks [18] in deep learning literature. Specifically, the dot-product attention scores between every feature pairs can be regarded as the importance of each
feature pair. Then, we can compute the second-order feature interaction as weighted sum over feature embeddings. To model arbitrary-order feature interaction, we can stack multiple layers of self-attentive networks with residual connections.

Intuitively, the dot product between each pair of feature embedding vectors encodes *pairwise* semantics of feature interaction, which, however, neglects the modeling of *general* influence of each feature field. To explicitly model such *unary* semantics, we propose to decouple a unary term from the vanilla self-attention network that computes the *general* impact of one certain feature to all other features. We term the resulting framework Disentangled Self-atTenTive Network, DESTINE for brevity. In particular, to better model feature interaction, DESTINE consists of two independent computational blocks illustrated in Figure 1: a *whitened* pairwise term that models *specific* interaction between two features and a unary term for the *general* influence of one feature to all others.

For the CTR prediction problem, we first embed input features into low-dimensional spaces and then compute high-order feature interaction by stacking multiple disentangled self-attentive layers. Finally, the embeddings resulting from the last interaction layer are used to estimate the click behavior. Extensive experiments on two real-world datasets demonstrate that our proposed DESTINE not only achieves state-of-the-art performance but also retains high computational efficiency. Our code is made publicly available at https://github.com/CRIPAC-DIG/DESTINE.

## 2 THE PROPOSED DESTINE APPROACH

### 2.1 Problem Definition

Suppose the training dataset $\mathbb{D} = \{x_i, y_i\}_{i=1}^N$ contains $N$ samples, where each sample $x_i$ consists of $M$ fields of users’ and items’ features and its associated label $y_i \in \{0, 1\}$ represents that user’s behavior (e.g., whether to click an item). The problem of click-through rate prediction is to predict $\hat{y}_i$, given a feature vector $x_i$, for accurately estimating whether a user will interact with an item.

### 2.2 Learning Decoupled Feature Interaction

The proposed DESTINE consists of three key components: (a) the embedding layer, (b) the interaction layer, and (c) the output layer. At first, the input features are fed into the embedding layer, which transforms input features into dense, low-dimensional embedding vectors. Then, these feature embeddings are fed into several stacked interaction layers, which model high-order interaction. After that, we feed the embeddings from the last interaction layer into the output layer to estimate the click behavior.

For each sparse input feature $x_i$, we transform it into dense embeddings $e_i \in \mathbb{R}^d$ via embedding lookup. Once we obtained a compact representation for each feature, we use the scaled dot-product attention scheme to model high-order feature interaction among feature fields. Specifically, we formulate each feature interaction as a (*key*, *value*) pair and learn the importance of each feature interaction by multiplying each feature embedding, such that important key–value pairs get higher attention scores. Formally, we first transform each feature embedding into a new embedding space $\mathbb{R}^d$ as follows

$$q_m = W_q e_m.$$  \hspace{1cm} (1)

where the *query* and *key* vectors are parameterized by two linear transformation matrices $W_q, W_k \in \mathbb{R}^{d \times d}$, respectively.

Then, we compute the correlation $\alpha(e_m, e_n)$ between feature $m$ and feature $n$.

For the pairwise term, we perform whitening [3] on the key and the query vector to model pure interaction among features, which makes the two interacting features less correlated with each other:

$$\alpha_q(e_m, e_n) = \sigma \left( (q_m - \mu_q)^\top (k_n - \mu_k) \right),$$ \hspace{1cm} (4)

where $\sigma(\cdot)$ is the softmax function; $\mu_q = \frac{1}{M} \sum_{i=1}^M W_q e_i$ and $\mu_k = \frac{1}{M} \sum_{i=1}^M W_k e_i$ takes average of the key and the query vectors, respectively.

Regarding the unary term, we introduce another query transformation matrix $W'_q \in \mathbb{R}^{d \times d}$ for modeling significant features:

$$\alpha_u(e_m, e_n) = \sigma \left( (\mu'_q)^\top k_n \right),$$ \hspace{1cm} (5)

where $\mu'_q$ is the mean vector from another transformation to model the general impact on key vectors, i.e., $\mu'_q = \frac{1}{M} \sum_{i=1}^M W'_q e_i$.

After computing the attention score for each feature interaction $(m, n)$, we transform each candidate feature to a new embedding space by a *value* transformation, parameterized by a linear transformation matrix $W_v \in \mathbb{R}^{d \times d}$, as given in the sequel,

$$v_k = W_v e_k.$$ \hspace{1cm} (6)

At last, we update the final representation for feature field $m$ by linearly combining all features.

**Extension to multihead self-attention.** To allow the model to learn distinct feature interaction in different subspaces, we use multiple attention heads. Specifically, we compute the representation of feature field $m$ under an attention head $h$ by

$$z_m^{(h)} = \sum_{k=1}^M \alpha_k^{(h)}(e_m, e_k) v_k^{(h)},$$ \hspace{1cm} (7)

where $\alpha_k^{(h)}$ is the attention score computed via Eq. (3). Note that, each attention head $h$ keeps its distinct weight parameters $W_k^{(h)}, W_q^{(h)}, (W_q')^{(h)},$ and $W_v^{(h)}$.

Then, we obtain $z_m$, the overall hidden representation for feature $m$, by concatenating the representation of all attention heads as

$$z_m = \left[ z_m^{(1)}, z_m^{(2)}, \ldots, z_m^{(H)} \right],$$ \hspace{1cm} (8)

where $H$ is the number of attention heads. Additionally, following previous work [16, 23], we incorporate raw, individual features (first-order features) via residual connections [5], which is formulated as,

$$\hat{z}_m = \psi(z_m + W_e e_m),$$ \hspace{1cm} (9)

where $W_e \in \mathbb{R}^{d'H \times d}$ is a linear projection matrix to avoid dimension mismatch and $\psi(\cdot) = \max(0, \cdot)$ is the ReLU activation function.
With features \( \hat{z}_m \) received from the last interaction layer, we concatenate all \( M \) features and utilize a simple logistic regression model on top of them to predict user behavior, formulated as
\[
\hat{y} = \sigma (w^T [\hat{z}_1; \hat{z}_2; \ldots; \hat{z}_M] + b),
\]
where \( w \in \mathbb{R}^{dH} \) is a linear projection vector, \( b \) is a bias term, and \( \sigma(x) = 1/(1 + e^{-x}) \) is the sigmoid function.

At last, we employ the binary cross-entropy function, which is widely-used in CTR prediction models, as the loss function,
\[
L = -\frac{1}{N} \sum_{(y, \hat{y}) \in D} (y \log \hat{y} + (1 - y) \log(1 - \hat{y})),
\]
where \( y \) and \( \hat{y} \) is the ground truth and the predicted click, respectively. We use gradient descent algorithms to update model weights.

2.3 Complexity Analysis
For the proposed model, we only introduce a new learnable parameter \( W_q^i \in \mathbb{R}^{d \times d'} \) for each attention head, leading to additional space complexity of \( O(d^2H) \) for each interaction layer. In addition, the time complexity of DESTINE is \( O(H(d^2M + d^2M + d'H^2M^2 + d'M^2)) = O(Md'H^2(M + d)) \), compared to that of the original self-attention module of \( O(Md'H^2(M + d)) \). Note that \( d', d, \) and \( H \) are usually small, demonstrating that our model is memory-efficient and computation-friendly.

3 EXPERIMENTS
In this section, we present empirical analysis to answer the following three questions.

RQ1. Does the proposed DESTINE method outperform existing state-of-the-art baseline methods that model feature interaction for CTR prediction?

RQ2. Many models integrate implicit feature interaction via deep neural networks (DNN); how does the proposed model with implicit feature interaction compare with them?

RQ3. How do other model variants perform compared to the proposed DESTINE?

3.1 Experimental Setup
We use two large-scale datasets Avazu and Criteo for evaluation. Avazu contains click logs of mobile ads in 10 days, which is composed of 23 categorical features including domains, categories, connection types, etc. Criteo comprises traffic logs of display ads over 7 days. Each sample contains 13 numerical and 26 categorical feature fields. The detailed statistics is summarized in Table 1.

For fair comparison, following most existing studies [9, 16, 24], we randomly sample 80% data as the training set, 10% as the test set, and the remaining 10% as the validation set. We report the performance in terms of two widely-used metrics AUC and logloss. Please kindly note that considering a large scale of user base, performance improvements of AUC at 1%-level are considered as practically significant for industrial deployment [2, 4, 16, 19, 20, 24].

3.2 Performance of Feature Interaction Models
3.2.1 Baselines. Representative baseline CTR prediction models can be grouped into three lines, according to the order of feature interaction they model: (a) first-order method LR [13], (b) second-order methods FM [12] and AFM [21], and (c) higher-order methods DeepCrossing [14], CrossNet [19], CIN [10], HOFM [1], and AutoInt [16]. In this section, we only include counterparts that directly model feature interaction; for full models that involve a DNN component to model implicit interaction, such as Deep&Cross and AutoInt+, the results are presented later in Section 3.3.

3.2.2 Implementation details. We set the number of attention heads to \( H = 2 \); the hidden dimension of embeddings is set to \( d' = 32 \) and the size of attentive embeddings is set to \( d = 64 \). The model is trained using the Adam optimizer [8] with a learning rate of 0.001. We find that the weight for \( l_2 \) regularization slightly impacts the model performance, and thus we carefully tuned it in \( [5 \times 10^{-3}, 5 \times 10^{-4}, 5 \times 10^{-5}, 5 \times 10^{-6}] \). Moreover, we set the dropout [17] rate to 0.2 to avoid overfitting.

3.2.3 Results and analysis. The overall performance is summarized in Table 2. We also report the total training time (mins) of each model. All baseline performance is referenced from their original papers. The training time of baselines is measured using their official implementations. Overall, from the table, it is evident that the proposed DESTINE achieves the best performance on all datasets. Moreover, DESTINE also enjoys another merit of low computational complexity as it takes relatively low time to finish training.

We also make other observations. Firstly, methods that model higher-order feature interaction generally gain larger performance improvements, which verifies the necessity of modeling high-order interaction in CTR prediction. Secondly, although several high-order methods such as DeepCrossing utilize a feedforward neural network to learn feature interaction, their performance is even inferior to second-order methods, which implies that it is not sufficient to learn useful feature interaction in an implicit way. Our method,
We observe that DESTINE with implicit feature interaction with DNN, we term our hybrid model DESTINE-P achieving new state-of-the-art performance on the contrary, explicitly models useful feature interaction via attentive networks, so that achieves better performance. Thirdly, compared to AutoInt that leverages a vanilla attentive net, our proposed DESTINE disentangles a pairwise and a unary term from computing attention scores, which further boosts performance. In summary, the results validate the effectiveness of DESTINE.

### 3.3 Performance of Deep Models Integrated

#### 3.3.1 Evaluation protocols

Many existing CTR prediction models alternatively integrate a Deep Neural Network (DNN) component to learn implicit feature interaction. For each DNN layer, we use a simple feedforward network, followed by a batch normalization layer [6]. We stack two DNN layers for all datasets. The resulting representation \( \tilde{z}_m \) for feature \( m \) is concatenated with \( \tilde{z}_m \) and is further trained with another linear layer to obtain the final embedding. For investigating whether implicit feature interaction improves the performance, we include the following baselines: Wide&Deep [2], DeepFM [4], Deep&Cross [19], xDeepFM [10], AutoInt+ [16], DeepIM [23], and AutoCTR [15]. Following the naming convention, we term our hybrid model DESTINE+ that jointly learns feature interaction using decoupled attentive networks and DNNs.

#### 3.3.2 Results and analysis

The results are summarized in Table 3. We observe that DESTINE+ achieves new state-of-the-art performance over existing baselines, which demonstrates that integrating with implicit feature interaction with DNN, DESTINE is able to learn feature interaction more effectively. Moreover, after the last two columns, we observe that the average improvements brought by the DNN component are limited. This indicates that the base model DESTINE has already achieved relatively high performance, which once again verifies the efficacy of DESTINE. We also note that the relative improvements on the Criteo dataset are greater than that on the Avazu. This could be explained by that the number of feature fields on Criteo is much more than that on Avazu; in other words, Criteo is “wider” than Avazu, where DNN training is much easier than the proposed disentangled self-attentive model.

### 3.4 Performance of Model Variants

For diagnosing the proposed scheme, we further investigate four variants of the proposed disentangled self-attention module:

- **Pairwise (DESTINE-P)**. We only use the pairwise term in \( Eq. \ (4) \).
- **Unary (DESTINE-U)**. We only use the unary term in \( Eq. \ (5) \).
- **Multiplication (DESTINE-M)**. We use multiplication rather than addition for combining the pairwise and the unary term, as described below,

\[
\alpha_d (\epsilon_m, \epsilon_n) = \sigma \left( (q_m - \mu_q) \top (k_n - \mu_k) \right) \sigma \left( (\mu_q') \top k_n \right).
\]

- **Shared transformation (DESTINE-S)**. We consider preserving the shared key transformation \( W_k \) in the unary term, as formulated below,

\[
\alpha_s (\epsilon_m, \epsilon_n) = \sigma \left( (q_m - \mu_q) \top (k_n - \mu_k) \right) + \sigma \left( \mu_q' k_n \right).
\]

The results are summarized in Figure 2. We can see that both the pairwise and the unary term benefits model performance. However, the way to combine the two terms plays an important role. DESTINE-M performs poorly despite its utilization of both terms and is even outperformed by DESTINE-U and DESTINE-P, where only one of the two terms is used. This verifies our justification that coupled gradient in DESTINE-M will deteriorate the performance. The performance DESTINE-S is also inferior to DESTINE, since the shared \( W_q \) transformation still leads to the coupling of gradients during learning. The outstanding performance of DESTINE compared to all other variants justifies the design of our model.

### 4 CONCLUSION

In this paper, we present a disentangled self-attention network DESTINE for click-through rate prediction, which consists of two terms for pairwise and unary semantics. Specifically, the unary term models the general impact of one feature on all others, whereas the remaining whitened pairwise term models pure feature interaction among each feature pair. Extensive experiments on two real-world datasets demonstrate the effectiveness of the proposed method.

### ACKNOWLEDGMENTS

This work is supported by National Key Research and Development Program (2018YFB1402600), National Natural Science Foundation of China (61772528), and Shandong Provincial Key Research and Development Program (2019JZZY010119).
REFERENCES

[1] Mathieu Blondel, Akinori Fujino, Naonori Ueda, and Masakazu Ishihata. 2016. Higher-Order Factorization Machines. In NIPS. 3351–3359.

[2] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg S. Corrado, Wei Chai, Mustafa Ispir, Rohan Amil, Zakaria Hague, Lichan Hong, Viyan Jain, Xiaobing Liu, and Hemal Shah. 2016. Wide & Deep Learning for Recommender Systems. In DLRS@RecSys. ACM, 7–10.

[3] Jerome H. Friedman. 1987. Exploratory Projection Pursuit. J. Amer. Statist. Assoc. 82, 397 (March 1987), 249–266.

[4] Huifeng Guo, Ruiming Tang, Yuming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. In IJCAI. 1725–1731.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In CVPR. IEEE, 770–778.

[6] Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In ICML. 448–456.

[7] Wang-Cheng Kang and Julian McAuley. 2018. Self-Attentive Sequential Recommendation. In ICDM. IEEE, 197–206.

[8] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In ICLR.

[9] Zekun Li, Zeyu Cui, Shu Wu, Xiaoyu Zhang, and Liang Wang. 2019. Fi-GNN: Modeling Feature Interactions via Graph Neural Networks for CTR Prediction. In CIKM. ACM, 539–548.

[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 KDD. ACM, 1754–1763.

[11] Ying Shan, T. Ryan Hoens, Jian Jiao, Huijing Wang, Dong Yu, and J C Mao. 2016. Deep Crossing: Web-Scale Modeling without Manually Crafted Combinatorial Features. In KDD. ACM, 255–262.

[12] Qingshan Song, Dehua Chen, Hanning Zhou, Jiyan Yang, Yuanzhong Tian, and Xia Hu. 2020. Towards Automated Neural Interaction Discovery for Click-Through Rate Prediction. In KDD. ACM, 945–955.

[13] Weiping Song, Chence Shi, Zhiqiang Zhao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. 2019. Autolint: Automatic Feature Interaction Learning via Self-Attentive Neural Networks. In CIKM. ACM, 1161–1170.

[14] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks From Overfitting. JMLR 15, 1 (2014), 1929–1958.

[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Urs Shuker, and Illia Polosukhin. 2017. Attention is All You Need. In NIPS. ACM, 5998–6008.

[16] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & Cross Network for Ad Click Predictions. In AdKDD@KDD. ACM, 12:1–12:7.

[17] Shu Wu, Feng Yu, Xueli Yu, Qiang Liu, Liang Wang, Tienniu Tan, Jie Shao, and Fan Huang. 2020. TPNet: Multi-Semantic Feature Interaction for CTR Prediction. In SIGIR. ACM, 1885–1888.

[18] 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. In IJCAI. 3119–3125.

[19] Minghao Yin, Zhuliang Yao, Yue Cao, Xiu Li, Zheng Zhang, Stephen Lin, and Han Hu. 2020. Disentangled Non-Local Neural Networks. In ECCV. Springer, 191–207.

[20] Jerming Zhu, Jinyang Liu, Shuai Yang, Qi Zhang, and Xiuqiang He. 2020. FuxiCTR: An Open Benchmark for Click-Through Rate Prediction. arXiv (Sept. 2020). arXiv:2009.05794v1 [cs.IR]