

# Supplementary Materials: Mitigating Sample Selection Bias with Robust Domain Adaption in Multimedia Recommendation

Anonymous Authors

## 1 MORE EXPERIMENTAL DETAILS

### 1.1 Implementation of Testbed

We construct a two-stage cascade recommendation system, comprising retrieval and ranking stages, as the testbed for performance comparison. The implementation of the cascade system refers to two popular frameworks, i.e., DeepMatch<sup>1</sup> and DeepCTR<sup>2,3</sup>. In the retrieval stage, we employ Deep Structured Semantic Model (DSSM) as the backbone, a widely used dual-tower structural model in industrial scenarios. This model processes user and item features separately, leveraging their similarity dot product to generate prediction results. For the user tower and the item tower, we both use a three-layer MLP with hidden dimensions of 300, 300, and 128 to obtain feature embeddings. In the ranking stage, we utilize two common CTR models, namely Deep Factorization-Machine (DeepFM) and Deep & Cross Network (DCN). DeepFM combines DNN and factorization machines to capture high-order and low-order feature interactions simultaneously. DCN incorporates the cross-network to make the degree of cross-features grow with layer depth. The DNN used in DeepFM and DCN consists of a two-layer MLP with hidden dimensions of 256 and 128.

### 1.2 Supplementary Details of DAMCAR

Our implementation code is available at an anonymous GitHub link: <https://anonymous.4open.science/r/DAMCAR-3367>. In generating robust pseudo-labels, we use Gradient Reversal Layer (GRL) to reverse gradients for alignment with the update direction. Specifically, GRL acts as an identity transformation during forward propagation, while in backward propagation, it reverses gradients from subsequent layers. This process can be formulated as follows:

$$\text{Forward : } \text{GRL}(g(z^s)), \text{GRL}(z^t) = g(z^s), z^t, \quad (1)$$

$$\text{Backward : } \frac{\partial \mathcal{L}_{dom}}{\partial g(z^s)}, \frac{\partial \mathcal{L}_{dom}}{\partial z^t} = -\beta \frac{\partial \mathcal{L}_{dom}}{\partial \text{GRL}(g(z^s))}, -\beta \frac{\partial \mathcal{L}_{dom}}{\partial \text{GRL}(z^t)}, \quad (2)$$

where the reverse ratio  $\beta$  is set to 1. Grid search is employed to determine the optimal weights of  $\mathcal{L}_{sup}$ ,  $\mathcal{L}_{dom}$ , and  $\mathcal{L}_{con}$ , resulting in  $\lambda_1 = 0.5$ ,  $\lambda_2 = 0.2$ , and  $\lambda_3 = 0.3$  on WeChat, while  $\lambda_1 = 0.5$ ,  $\lambda_2 = 0.3$ , and  $\lambda_3 = 0.4$  on TikTok.

## 2 ADDITIONAL ABLATION STUDY

In this section, we explore the impact of specific design details on recommendation performance. In generating robust pseudo-labels, we introduce a linear transformation  $g(\cdot)$  to ease the challenge of aligning source and target domain distributions. Table 1 presents the ranking performance comparison. Moreover, we integrate the scores  $y_{rank}$  obtained from the ranking model as one of the training

**Table 1: Ablation Study of the linear transformation  $g(\cdot)$  on WeChat and TikTok, using DSSM and DeepFM as the backbone models. The best results are marked in bold.**

Method	WeChat			TikTok		
-	N@10	M@10	H@10	N@10	M@10	H@10
DAMCAR	<b>0.0657</b>	<b>0.0394</b>	<b>0.0166</b>	<b>0.0443</b>	<b>0.0356</b>	<b>0.0089</b>
w/o $g(\cdot)$	0.0646	0.0387	0.0165	0.0431	0.0350	0.0087

**Table 2: Ablation Study of the ranking scores  $y_{rank}$  on WeChat and TikTok, using DSSM and DeepFM as the backbone models. The best results are marked in bold.**

Method	WeChat			TikTok		
-	N@10	M@10	H@10	N@10	M@10	H@10
DAMCAR	<b>0.0657</b>	<b>0.0394</b>	<b>0.0166</b>	<b>0.0443</b>	<b>0.0356</b>	<b>0.0089</b>
w/o $y_{rank}$	0.0602	0.0361	0.0159	0.0420	0.0336	0.0081

**Table 3: Performance comparison of the retrieval model under different methods. The best and second-best results are marked in bold and underlined, respectively.**

Method	WeChat		TikTok	
-	AUC $\uparrow$	LL $\downarrow$	AUC $\uparrow$	LL $\downarrow$
BC	0.6897	0.6412	0.6517	0.6594
KD	0.6988	0.6382	0.6692	0.6564
TL	0.7055	0.6334	0.6605	0.6571
AR	<u>0.7060</u>	<u>0.6333</u>	0.6696	0.6535
MUDA	0.7017	0.6369	<u>0.6748</u>	<u>0.6516</u>
DAMCAR	<b>0.7095</b>	<b>0.6320</b>	<b>0.6801</b>	<b>0.6440</b>

objectives to ensure consistency in the cascade system. Table 2 compares two variants, with and without utilizing  $y_{rank}$ .

The experimental results demonstrate that both the introduction of the linear transformation and the ranking scores contribute to enhancements in the final recommendation performance.

## 3 COMPARISON IN MORE METRICS

To compare the performance of the retrieval model under different methods more comprehensively, we add two metrics, namely Area Under the ROC Curve (AUC) and Log Loss (LL), which are widely used in evaluating classification ability. Calculations are performed using the popular machine learning framework, i.e., scikit-learn. Note that for AUC, higher values indicate better results, while for LL, lower values indicate better results. Table 3 illustrates that DAMCAR outperforms other baselines in both metrics.

<sup>1</sup><https://github.com/bbruceyuan/DeepMatch-Torch>

<sup>2</sup><https://github.com/shenweichen/DeepCTR-Torch>

<sup>3</sup><https://github.com/MemoryForSky/deepctr>