A Related Work

We remind important related works to understand how our AdvInfoNCE stands and its role in rich literature. Our work is related to the literature on contrastive learning-based collaborative filtering (CL-based CF) methods, and theoretical understanding of contrastive loss in collaborative filtering.

A.1 Contrastive Learning-based Collaborative Filtering

The latest **CL-based CF methods** can roughly fall into two research lines. The first one, which we term the "augmentation-based" approach, leverages user-item bipartite graph augmentations along with augmented views as positive signals. The second category, referred to as "loss-based" approaches, mainly focuses on the modification of contrastive loss. In loss-based CF models, interacted items serve as positive instances.

- **Augmentation-based [13–15, 52, 26, 27, 63–66].** The prevailing augmentation-based paradigm in CL-based CF methods is to employ user-item bipartite graph augmentations to generate contrasting views. These contrasting views are then treated as positive instances in the application of contrastive loss, such as InfoNCE loss, to further enhance collaborative filtering signals. Recent studies have extensively explored methods for generating contrastive views. Several studies like SGL [13] and DCL [64] elaborate on data-heuristic augmentation operators such as a random node or edge dropout and random walk. NCL [52] takes a different approach and incorporates potential neighbors from both the graph structure and semantic space into contrastive views. XSimGCL [27] takes it a step further and discards ineffective graph augmentations, choosing instead to employ a simple noise-based embedding augmentation. In pursuit of high-quality augmented supervision signals instead of handcrafted strategies, AutoCF [15] designs a learnable masking function to automatically identify important centric nodes for data augmentation.
- **Loss-based [53–55, 22, 67].** Recent research, such as the experiments presented in SimGCL [26] and XSimGCL [27], has empirically shown that contrastive loss can be instrumental in enhancing the performance of CF methods, often playing a more significant role than heuristic-based graph augmentation. Despite these findings, there remains a gap in the exploration of loss-based CF methods, an area ripe for further investigation. BC loss [54] incorporates bias-aware margins into the contrastive loss, enabling the learning of high-quality head and tail representations with robust discrimination and generalization abilities. Adap-τ [55] proposes an adaptive fine-grained strategy for selecting the personalized temperature τ for each user within the contrastive loss. HDCCF [22] devises a new contrastive loss function extending the advantage of negative mining from user-item to neighbored users and items.

A.2 Theoretical Understanding of Contrastive Loss in CF

Despite the remarkable success of CL-based CF methods, there remains a lack of theoretical understanding, particularly regarding the superior generalization ability of contrastive loss. In a study conducted by [68], three model-agnostic advantages of contrastive loss are theoretically revealed, including mitigating popularity bias, mining hard negative samples, and maximizing the ranking metric. CLRec [14] sheds light on contrastive loss from a bias-reduction perspective by revealing its connection with inverse propensity weighting techniques. XSimGCL [27] suggests that contrastive learning enables the recommender to learn more evenly distributed user and item representations, thereby mitigating the prevalent popularity bias in CF.

B In-depth Analysis of AdvInfoNCE

B.1 Complete Derivation of AdvInfoNCE

Full Derivation. We first introduce the widely-used LogSumExp operator in machine learning algorithms.

$$\max(x_1, x_2, ..., x_n) \approx \log(\exp(x_1) + \exp(x_2) + ... + \exp(x_n)) \tag{10}$$

The fine-grained ranking criterion for a single positive interaction (u, i) is defined as:

$$\forall j \in \mathcal{N}_u, \quad s(u, j) - s(u, i) + \delta_j < 0. \tag{11}$$

Then we probe into the Eq (11) and transform it into an optimization problem as follows:

$$\min_{\theta} \max\{0, \{s(u,j) - s(u,i) + \delta_j\}_{j\in\mathcal{N}_u}\} \tag{12}$$

We can seamlessly transform this optimization objective into the core component of our AdvInfoNCE:

$$\begin{aligned}
\underbrace{\max\{0, \{s(u,j) - s(u,i) + \delta_j\}_{j\in\mathcal{N}_u}\}}_{\text{Hardness-aware ranking criterion}} &\approx \log(\exp(0) + \sum_{j=1}^{|\mathcal{N}_u|} \exp(s(u,j) - s(u,i) + \delta_j)) \\
&= \log\{1 + \sum_{j=1}^{|\mathcal{N}_u|} \exp(\delta_j)\exp(s(u,j) - s(u,i))\} \\
&= \log\{1 + |\mathcal{N}_u| \sum_{j=1}^{|\mathcal{N}_u|} \frac{\exp(\delta_j)}{|\mathcal{N}_u|} \exp(s(u,j) - s(u,i))\} \\
&= \underbrace{-\log\{\frac{\exp(s(u,i))}{\exp(s(u,i)) + |\mathcal{N}_u| \sum_{j=1}^{|\mathcal{N}_u|} \frac{\exp(\delta_j)}{|\mathcal{N}_u|} \exp(s(u,j))}\}}_{\text{AdvInfoNCE}}
\end{aligned} \tag{13}$$

Drawing inspiration from adversarial training [48], we utilize a min-max game that allows for alternating training of the model between predicting hardness and refining the CF model. Formally, we formulate the AdvInfoNCE learning framework as the following optimization problem:

$$\min_{\theta} \mathcal{L}_{\text{AdvInfoNCE}} = \min_{\theta} \max_{\Delta\in\mathbb{C}(\eta)} - \sum_{(u,i)\in\mathcal{O}^+} \log \frac{\exp(s(u,i))}{\exp(s(u,i)) + |\mathcal{N}_u| \sum_{j=1}^{|\mathcal{N}_u|} \frac{\exp(\delta_j^{(u,i)})}{|\mathcal{N}_u|} \exp(s(u,j))} \tag{14}$$

where $\frac{\exp(\delta_j^{(u,i)})}{|\mathcal{N}_u|} \in \mathbb{C}(\eta, (u,i)) = (\frac{1}{|\mathcal{N}_u|} - \epsilon, \frac{1}{|\mathcal{N}_u|} + \epsilon)$, and ϵ is a hyperparameter that regulates the upper-bound deviation of hardness. In practice, ϵ is regulated by the number of adversarial training epochs under a fixed learning rate (refer to Algorithm 1).

If we further define $\frac{\exp(\delta_j^{(u,i)})}{|\mathcal{N}_u|}$ as $p(j|(u,i))$, which signifies the likelihood of selecting item j as a negative sample for the user-item pair (u,i), we can rewrite the AdvInfoNCE loss as:

$$\min_{\theta} \mathcal{L}_{\text{AdvInfoNCE}} = \min_{\theta} \max_{p(\cdot|\cdot)} - \sum_{(u,i)\in\mathcal{O}^+} \log \frac{\exp(s(u,i))}{\exp(s(u,i)) + |\mathcal{N}_u| \sum_{j=1}^{|\mathcal{N}_u|} p(j|(u,i)) \exp(s(u,j))} \tag{15}$$

□

B.2 Proof of Theorem

Theorem 3.1. *We define* $\delta_j^{(u,i)} \doteq \log(|\mathcal{N}_u| \cdot p(j|(u,i)))$*, where* $p(j|(u,i))$ *is the probability of sampling negative item* j *for observed interaction* (u,i)*. Then, optimizing AdvInfoNCE loss is* ***equivalent*** *to solving Kullback-Leibler (KL) divergence-constrained distributionally robust optimization (DRO) problems over negative sampling:*

$$\min_{\theta} \mathcal{L}_{\textit{AdvInfoNCE}} \iff \min_{\theta} \max_{p(j|(u,i))\in\mathbb{P}} \mathbb{E}_P[\exp(s(u,j) - s(u,i)) : \mathcal{D}_{KL}(P_0||P) \leq \eta] \tag{8}$$

where P_0 *stands for the distribution of uniformly drawn negative samples,* i.e., $p_0(j|(u,i)) = \frac{1}{|\mathcal{N}_u|}$*;* P *denotes the distribution of negative sampling* $p(j|(u,i))$*.*

Proof. We denote the relative hardness of negative item j with respect to observed interaction (u,i) as $\delta_j^{(u,i)}$ and redefine it as $\log(|\mathcal{N}_u| \cdot p(j|(u,i)))$. In this definition, $|\mathcal{N}u|$ is the number of negative samples for each user u, and $p(j|(u,i))$ is the probability of selecting item j as a negative sample for a given user-item pair (u,i).

With the constraint that $\sum_{j=1}^{|\mathcal{N}_u|} p(j|(u,i)) = 1$, we can recalculate the average hardness as follows:

$$\begin{aligned}\frac{1}{|\mathcal{N}_u|}\sum_{j=1}^{|\mathcal{N}_u|}\delta_j^{(u,i)} &= \frac{1}{|\mathcal{N}_u|}\sum_{j=1}^{|\mathcal{N}_u|}\log(|\mathcal{N}_u|\cdot p(j|(u,i))) \\ &= -\sum_{j=1}^{|\mathcal{N}_u|}\frac{1}{|\mathcal{N}_u|}\log(\frac{1}{|\mathcal{N}_u|}\cdot\frac{1}{p(j|(u,i))}) \\ &= -\mathcal{D}_{KL}(P_0||P) \end{aligned} \tag{16}$$

In this formulation, P_0 represents the distribution of uniformly drawn negative samples, where $p_0(j|(u,i)) = \frac{1}{|\mathcal{N}_u|}$. Meanwhile, P denotes the distribution of negative sampling, represented as $p(j|(u,i))$.

Since $p(j|(u,i)) \in \mathbb{C}(\eta,(u,i)) = (\frac{1}{|\mathcal{N}_u|}-\epsilon, \frac{1}{|\mathcal{N}_u|}+\epsilon)$, we further define $\eta = -log(1-\frac{\epsilon^2}{|\mathcal{N}_u|})$. Thus, the feasible zone presented above is equivalent to a relaxed constraint $\mathcal{D}_{KL}(P_0||P) \leq \eta$.

The AdvInfoNCE loss for all observations can be rewritten in a different form by reorganizing and simplifying the Eq (15):

$$\begin{aligned}\min_\theta \mathcal{L}_{\text{AdvInfoNCE}} &= \min_\theta \sum_{(u,i)\in\mathcal{O}^+}\max_{p(j|(u,i))\in\mathbb{P}}\log\{1+|\mathcal{N}_u|\sum_{j=1}^{|\mathcal{N}_u|}p(j|(u,i))\exp(s(u,j)-s(u,i))\} \\ &\Longleftrightarrow \min_\theta \sum_{(u,i)\in\mathcal{O}^+}\max_{p(j|(u,i))\in\mathbb{P}}\sum_{j=1}^{|\mathcal{N}_u|}p(j|(u,i))\exp(s(u,j)-s(u,i)) \\ &\Longleftrightarrow \min_\theta \sum_{(u,i)\in\mathcal{O}^+}\sup_{p(j|(u,i))\in\mathbb{P}}\mathbb{E}_P[\exp(s(u,j)-s(u,i))] \end{aligned} \tag{17}$$

The equation presented above exemplifies a widely encountered formulation of the Distributionally Robust Optimization (DRO) problem [49] where the ambiguity set of the probability distribution is defined by the Kullback-Leibler (KL) divergence $\mathcal{D}_{KL}(P_0||P) \leq \eta$. □

B.3 Gradients Analysis

In this section, we delve into the crucial role of hardness δ_j in controlling the penalty strength on hard negative samples. The analysis begins with the main part of AdvInfoNCE for a single positive interaction (u,i), as defined in Eq (13), primarily due to its simplicity. For the sake of notation simplicity, let us denote it as:

$$\mathcal{L}_{\text{Adv}}(u,i) = -\log\{\frac{\exp(s(u,i))}{\exp(s(u,i))+\sum_{j=1}^{|\mathcal{N}_u|}\exp(\delta_j)\exp(s(u,j))}\} \tag{18}$$

Then the gradient with respect to the positive representations $\phi_\theta(i)$ of item i is formulated as:

$$\begin{aligned}-\nabla_i\mathcal{L}_{\text{Adv}}(u,i) &= \frac{\partial\mathcal{L}_{\text{Adv}}(u,i)}{\partial s(u,i)}\cdot\frac{\partial s(u,i)}{\partial\phi_\theta(i)} \\ &= (1-\frac{\exp(s(u,i))}{\exp(s(u,i))+\sum_{j=1}^{|\mathcal{N}_u|}\exp(\delta_j)\exp(s(u,j))})\cdot\frac{\phi_\theta(i)}{\tau} \end{aligned} \tag{19}$$

The gradients with respect to the negative representations $\phi_\theta(j)$ of item j is given by:

$$\begin{aligned}
-\nabla_j \mathcal{L}_{\text{Adv}}(u,i) &= \frac{\partial \mathcal{L}_{\text{Adv}}(u,i)}{\partial s(u,j)} \cdot \frac{\partial s(u,j)}{\partial \phi_\theta(j)} \\
&= \frac{\exp(\delta_j)\exp(s(u,j))}{\exp(s(u,i)) + \sum_{j=1}^{|\mathcal{N}_u|}\exp(\delta_j)\exp(s(u,j))} \cdot \frac{\phi_\theta(j)}{\tau} \\
&= \exp(\delta_j)\{1 - \frac{\exp(s(u,i))}{\exp(s(u,i)) + \sum_{j=1}^{|\mathcal{N}_u|}\exp(\delta_j)\exp(s(u,j))}\}\frac{\exp(s(u,j))}{\sum_{j=1}^{|\mathcal{N}_u|}\exp(\delta_j)\exp(s(u,j))}\frac{\phi_\theta(j)}{\tau}
\end{aligned} \tag{20}$$

Clearly, for a given user u, the gradient with respect to the positive item i equals the sum of gradients of all negative items, in accordance with the findings in [38]. The hardness $\exp(\delta_j)$ dictates the importance of negative gradients. Specifically, the gradients relating to the negative item j in Eq (20) correlate proportionally to the hardness term $\exp(\delta_j)$, which shows that the AdvInfoNCE loss function is hardness-aware.

B.4 Align Top-K evaluation metric

Discounted Cumulative Gain (DCG) is a commonly used ranking metric in top-K recommendation tasks. In DCG, the relevance of an item's contribution to the utility decreases logarithmically in relation to its position in the ranked list. This mimics the behavior of a user who is less likely to scrutinize items that are positioned lower in the ranking. Formally, DCG over rank $\pi_s(u,i)$ is defined as follows:

$$DCG(\pi_s(u,\mathcal{I}),\mathbf{y}) = \sum_{i=1}^{|\mathcal{I}|}\frac{2^{y_i}-1}{\log_2(1+\pi_s(u,i))} \tag{21}$$

Where $\pi_s(u,\mathcal{I})$ is a ranked list over $\mathcal{I}$, as determined by the similarity function s for user u, and $\mathbf{y}$ is a label vector that indicates whether an interaction has occurred previously or not. Then $\pi_s(u,i)$ is the rank of item i. Building on the research presented in [68], we explore how well AdvInfoNCE aligns with DCG for our purposes.

Under our proposed fine-grained hardness ranking criteria defined in Eq (11), the $\pi_s(u,i)$ can be obtained as follows:

$$\begin{aligned}
\pi_s(u,i) &= 1 + \sum_{j\in\mathcal{I}\setminus\{i\}} \mathbb{1}(s(u,j) - s(u,i) + \delta_j > 0) \\
&\leq 1 + \sum_{j\in\mathcal{I}\setminus\{i\}} \exp(s(u,j) - s(u,i) + \delta_j).
\end{aligned} \tag{22}$$

The last inequality is satisfied by $\mathbb{1}(x>0) \leq \exp(x)$.

$$\begin{aligned}
-\log[DCG(\pi_s(u,\mathcal{I}),\mathbf{y})] &= -\log\left[\sum_{i=1}^{|\mathcal{I}|}\frac{2^{y_i}-1}{\log_2(1+\pi_s(u,i))}\right] \\
&\leq -\log\left[\frac{1}{\log_2(1+\pi_s(u,i))}\right] \leq -\log\left[\frac{1}{\pi_s(u,i)}\right] \\
&\leq -\log(\frac{1}{1+\sum_{j\in\mathcal{I}\setminus\{i\}}\exp(s(u,j)-s(u,i)+\delta_j)}) \\
&= -\log(\frac{\exp(s(u,i))}{\exp(s(u,i))+\sum_{j\in\mathcal{I}\setminus\{i\}}\exp(s(u,j)+\delta_j)}) \\
&= \mathcal{L}_{\text{Adv}}(u,i) \leq \mathcal{L}_{\text{AdvInfoNCE}}
\end{aligned} \tag{23}$$

Suppose that there are K items in $\mathcal{I}$ that are interacted with u, let them to be $\{1,2,\cdots,K\}$ ithout loss of generality. Then

$$DCG(\pi_s(u,\mathcal{I}),\mathbf{y}) \leq \sum_{i=1}^{K}\frac{1}{\log_2(1+i)}, \tag{24}$$

the equality holds if and only if the interactions $\{(u, i) : i = 1, 2, \cdots, K\}$ are ranked top-K. For a given u,

$$\begin{aligned}
\sum_{i=1}^{K} \mathcal{L}_{\text{Adv}}(u, i) &= \sum_{i=1}^{K} -\log\left(\frac{\exp(s(u, i))}{\exp(s(u, i)) + \sum_{j \in \mathcal{I} \setminus \{i\}} \exp(s(u, j) + \delta_j)}\right) \\
&= \sum_{i=1}^{K} \log(1 + \sum_{j \in \mathcal{I} \setminus \{i\}} \exp(s(u, j) - \exp(s(u, i)) + \delta_j)) \\
&\geq \sum_{i=1}^{K} \log(1 + \sum_{j \in \mathcal{I} \setminus \{i\}} e^{-1}) = \sum_{i=1}^{K} \log(1 + (|\mathcal{I}| - 1)e^{-1}) \qquad (25) \\
&\geq \sum_{i=1}^{K} \frac{1}{\log_2(1 + i)} \geq DCG(\pi_s(u, \mathcal{I}), \mathbf{y}). \qquad (26)
\end{aligned}$$

The inequality in Eq (25) holds under the common usage of $s(u, i) \in [0, 1]$, the first inequality in Eq (26) holds when $|\mathcal{I}| \geq 6$, while the second inequality in Eq (26) holds by Eq (24).

Therefore, by Eq (23) and Eq (26),

$$\mathcal{L}_{\text{AdvInfoNCE}} \geq DCG(\pi_s(u, \mathcal{I}), \mathbf{y}) \geq \exp(-\mathcal{L}_{\text{AdvInfoNCE}}). \qquad (27)$$

Consequently, minimizing $\mathcal{L}_{\text{AdvInfoNCE}}$ is equivalent to minimizing $DCG(\pi_s(u, \mathcal{I}), \mathbf{y})$.

C Experiments

C.1 Experimental Settings

Datasets

- **KuaiRec** [56] is a real-world dataset sourced from the recommendation logs of KuaiShou, a platform for sharing short videos. The unbiased testing data consist of dense ratings from 1411 users for 3327 items, with the training data being relatively sparse. We categorize items that have a viewing duration exceeding twice the length of the short video as positive interactions.
- **Yahoo!R3** [57] is a dataset that encompasses ratings for songs. The training set is comprised of 311,704 user-selected ratings ranging from 1 to 5. The test set includes ratings for ten songs randomly exposed to each user. Interactions with items receiving a rating of 4 or higher are considered positive in our experiments.
- **Coat** [58] records online shopping interactions of customers purchasing coats. The training set, characterized as a biased dataset, comprises ratings provided by users for 24 items they have chosen. The test set, on the other hand, contains ratings for 16 coats that were randomly exposed to each user. Ratings in Coat follow a 5-point scale, and interactions involving items with a rating of 4 or above are classified as positive instances in our experiments.
- **Tencent** [10] is collected from the Tencent's short-video platform. We sort the items according to their popularity in descending order and divide them into 50 groups. Each group, defined by its popularity rank, is assigned a certain number of interactions, denoted by N_i, for inclusion in the test set. The quantity N_i is calculated based on $N_0 \cdot \gamma^{-\frac{i-1}{49}}$, where N_0 is the maximum number of interactions across all test groups and γ denotes the extent of the long-tail distribution. A lower value of gamma indicates a stronger deviation from the original distribution, thus yielding a more evenly distributed test set. To ensure that the validation set mirrored the long-tail distribution of

Table 3: Dataset statistics.

	KuaiRec	Yahoo!R3	Coat	Tencent
#Users	7,176	14,382	290	95,709
#Items	10,728	1,000	295	41,602
#Interactions	1,304,453	129,748	2,776	2,937,228
Density	0.0169	0.0090	0.0324	0.0007

Table 4: The performance comparison on unbiased datasets over the MF backbone. The improvement achieved by AdvInfoNCE is significant (p-value $<< 0.05$).

	Yahoo!R3		Coat	
	Recall	NDCG	Recall	NDCG
BPR (Rendle et al., 2012)	0.1189	0.0546	0.2782	0.1748
InfoNCE (van den Oord et al., 2018)	0.1478	0.0694	0.2683	0.1961
CCL (Mao et al., 2021)	0.1458$^{-1.35\%}$	0.0689$^{-0.72\%}$	0.2682$^{-0.04\%}$	0.1712$^{-12.70\%}$
BC Loss (Zhang et al., 2022)	0.1492$^{+0.95\%}$	<u>0.0698</u>$^{+0.58\%}$	0.2698$^{+0.56\%}$	0.1959$^{-0.10\%}$
Adap-τ (Chen et al., 2023)	<u>0.1512</u>$^{+2.30\%}$	0.0694$^{+0.43\%}$	<u>0.2712</u>$^{+1.08\%}$	<u>0.1986</u>$^{+1.27\%}$
AdvInfoNCE	**0.1523***$^{+3.04\%}$	**0.0710***$^{+2.31\%}$	**0.2905***$^{+8.27\%}$	**0.1999***$^{+1.94\%}$

the training set and no side information of the test set is leaked, the remaining interactions are randomly divided into training and validation sets at a ratio of 60:10.

Baselines. In this study, we conduct a comprehensive evaluation of AdvInfoNCE using two widely adopted collaborative filtering backbones, MF [50] and LightGCN [51]. We thoroughly compare AdvInfoNCE with two categories of the latest CL-based CF methods: augmentation-based baselines (SGL [13], NCL [52], XSimGCL [27]) and loss-based baselines (CCL [53], BC Loss [54], Adap-τ [55]).

- **SGL** [13] leverages data-heuristic graph augmentation techniques to generate augmented views. It then employs contrastive learning on the augmented views and the original embeddings.
- **NCL** [52] implements contrastive learning on two types of neighbors: structural neighbors and semantic neighbors. Structural neighbors are represented by the embeddings derived from even-numbered layers in a Graph Neural Network (GNN). Semantic neighbors comprise nodes with similar features or preferences, clustered through the Expectation-Maximization (EM) algorithm.
- **XSimGCL** [27] directly infuses noise into graph embeddings from the mid-layer of LightGCN to generate augmented views. This method arises from experimental observations indicating that CF models are relatively insensitive to graph augmentation. Instead, the key determinant of their performance lies in the application of contrastive loss.
- **CCL** [53] proposes a variant of contrastive loss based on cosine similarity. Specifically, it implements a strategy for filtering out negative samples lacking substantial information by employing a margin, denoted as m.
- **BC Loss** [54] integrates a bias-aware margin into the contrastive loss to alleviate popularity bias. Specifically, the bias-aware margin is learned via a specialized popularity branch, which only utilizes the statistical popularity of users and items to train an additional CF model.
- **Adap-τ** [55] proposes to automatically search the temperature of InfoNCE. Moreover, a fine-grained temperature is assigned for each user according to their previous loss.

Evaluation Metrics. We apply the all-ranking strategy, in which all items, with the exception of the positive ones present in the training set, are ranked by the collaborative filtering model for each user. An exception is KuaiRec, where the unbiased test set clusters in a small matrix [56]. This unique structure leads to a failure of the all-ranking strategy as the test set is not randomly selected from the whole user-item matrix. Consequently, for KuaiRec, we rank only the 3327 fully exposed items during the testing phase.

C.2 Performance over the MF Backbone

Given that the implementation of augmentation-based methods is tied to the LightGCN architecture, we compare AdvInfoNCE using the Matrix Factorization (MF) backbone against only loss-based methods. As shown in Table 4 and 5, AdvInfoNCE consistently surpasses all collaborative filtering baselines with modified contrastive losses. Moreover, similar to the trend observed with the LightGCN backbone, AdvInfoNCE excels on test sets exhibiting higher distribution shifts, while still preserving remarkable performance on the in-distribution validation set. The superior performance of AdvInfoNCE across both the LightGCN and MF backbones emphasizes its model-agnostic characteristic. We advocate for considering AdvInfoNCE as a standard loss in recommender systems.

Table 5: The performance comparison on the Tencent dataset over the MF backbone. The improvement achieved by AdvInfoNCE is significant (p-value $<<$ 0.05).

	γ = 200			γ = 10			γ = 2			Validation
	HR	Recall	NDCG	HR	Recall	NDCG	HR	Recall	NDCG	NDCG
BPR (Rendle et al., 2012)	0.0835	0.0299	0.0164	0.0516	0.0190	0.0102	0.0357	0.0141	0.008	0.0533
InfoNCE (van den Oord et al., 2018)	0.1476	0.0538	0.0318	0.0920	0.0334	0.0194	0.0627	0.0233	0.0141	0.0856
CCL (Mao et al., 2021)	0.1395	0.0523	0.0317	0.0930	0.0353	0.0221	0.0683	0.0266	0.0170	0.0782
BC Loss (Zhang et al., 2022)	<u>0.1546</u>	<u>0.0575</u>	<u>0.0349</u>	<u>0.1011</u>	<u>0.0378</u>	<u>0.0228</u>	<u>0.0737</u>	<u>0.0280</u>	<u>0.0178</u>	0.0864
Adap-τ (Chen et al., 2023)	0.1398	0.0512	0.0302	0.0876	0.0316	0.0182	0.0591	0.0221	0.0134	0.0844
AdvInfoNCE	**0.1606***	**0.0595***	**0.0355***	**0.1111***	**0.0412***	**0.0249***	**0.0813***	**0.0308***	**0.0189***	0.0860
Imp.% over the strongest baseline	3.87%	3.52%	1.86%	9.86%	8.86%	9.38%	10.33%	9.87%	6.28%	–
Imp.% over InfoNCE	8.84%	10.51%	11.77%	20.81%	23.52%	28.41%	29.64%	31.97%	33.69%	–

Table 6: The performance comparison on the Tencent dataset over extensive backbones. The improvement achieved by AdvInfoNCE is significant (p-value $<<$ 0.05).

	γ = 200			γ = 10			γ = 2			Validation
	HR	Recall	NDCG	HR	Recall	NDCG	HR	Recall	NDCG	NDCG
UltraGCN (Mao et al., 2021)	0.0930	0.0343	0.0190	0.0567	0.0215	0.0119	0.0400	0.0157	0.0095	0.0682
UltraGCN + InfoNCE	0.1436	0.0519	0.0303	0.0896	0.0324	0.0189	0.0617	0.0227	0.0135	0.0842
UltraGCN + AdvInfoNCE	<u>0.1538</u>	<u>0.0569</u>	<u>0.0338</u>	<u>0.1025</u>	<u>0.0380</u>	<u>0.0227</u>	<u>0.0726</u>	<u>0.0276</u>	<u>0.0168</u>	0.0883
VGAE (Kipf and Welling, 2016) + InfoNCE	0.1482	0.0536	0.0315	0.0923	0.0338	0.0202	0.0640	0.0237	0.0141	0.0823
VGAE + AdvInfoNCE	**0.1588***	**0.0589***	**0.0353***	**0.1069***	**0.0395***	**0.0239***	**0.0778***	**0.0296***	**0.0182***	0.0871

C.3 Performance over Extensive Backbones

To validate the generalization ability of AdvInfoNCE, we conducted experiments on additional backbones, including UltraGCN [69] and an adapted version of VGAE [70]. The results in Table 6 indicate that AdvInfoNCE performs excellently across various backbones, which showcases the generalization ability of AdvInfoNCE.

C.4 Performance Comparison with Extensive Baselines

We compare AdvInfoNCE on the LightGCN backbone with extensive baselines on Tencent. The results in Table 7 show that AdvInfoNCE also outperforms almost all the latest debiasing [10, 71] and hard negative mining algorithms [72].

C.5 Training Cost

Let n be the number of items, d be the embedding size, N be the number of negative sampling, M = $|\mathcal{O}^+|$ be the number of observed interactions, B be the batch size and N_b be the number of mini-batches within one batch. In AdvInfoNCE, the similarity calculation for one positive item with N negative items costs $O((N+1)d)$, and the hardness calculation costs $O(Nd)$. The total training costs of one epoch without backward propagation are summarized in Table 8. The training cost of AdvInfoNCE is a little higher than BPR loss, sharing the same complexity with InfoNCE.
In Table 9, we present both the per-epoch and total time costs for each baseline model on the Tencent dataset. As evidenced, augmentation-based contrastive learning (CL) baselines significantly cut down the overall training time, while loss-based CL baselines exhibit a complexity similar to that of InfoNCE. Surprisingly, compared to InfoNCE, AdvInfoNCE introduces only a marginal increase in computational complexity during the training phase.

D Discussion about AdvInfoNCE

D.1 Algorithm

Algorithm 1 depicts the detailed procedure of AdvInfoNCE. Here we uniformly sample N negative items for each observed interaction and multiply a large weighting parameter K in front of each negative item, as a surrogate of the whole negative set $\mathcal{N}_u$. Specifically, we adversarially train the hardness $\delta_j^{(u,i)}$ at a fixed interval before reaching the maximum adversarial training epochs E_{adv}. The precise methods for computing the hardness $\delta_j^{(u,i)}$ are further discussed in Section D.4.

Table 7: The performance comparison on the Tencent dataset with extensive baselines. The improvement achieved by AdvInfoNCE is significant (p-value $<<$ 0.05).

	γ = 200			γ = 10			γ = 2			Validation
	HR	Recall	NDCG	HR	Recall	NDCG	HR	Recall	NDCG	NDCG
XIR (Chen et al., 2022)	0.1463	0.0538	0.0326	0.0936	0.0341	0.0211	0.0642	0.0245	0.0154	0.0883
sDRO (Wen et al., 2022)	0.1455	0.0516	0.0286	0.0857	0.0304	0.0166	0.0552	0.0205	0.0110	0.0872
InvCF (Zhang et al., 2023)	**0.1651**	**0.0605**	0.0331	0.1061	0.0386	0.0204	0.0722	0.0272	0.0149	0.0912
AdvInfoNCE	0.1600	0.0594	**0.0356***	**0.1087***	**0.0403***	**0.0243***	**0.0774***	**0.0295***	**0.0180***	0.0879

Table 8: Time Complexity

	+BPR	+InfoNCE	+CCL	+BC Loss	+Adap τ	+AdvInfoNCE
Backbone	$O(N_bBd)$	$O(N_bB(N+1)d)$	$O(N_bB(N+1)d)$	$O(N_bB(N+1)d)$	$O(N_bB(N+1)d+(M+n)d)$	$O(N_bB(N+1)d)$

D.2 Effect of the Fine-grained Hardness on KuaiRec

We conduct the same experiments as Section 4.2.1 on KuaiRec, to investigate how the fine-grained hardness affects the out-of-distribution performance.

- We plot the average value of $p(j|(u,i))$ across one batch, as depicted in Figure 4a. The figure reveals that AdvInfoNCE learns a skewed negative sampling distribution, mirroring the trend observed in the Tencent dataset. Such a distribution places more emphasis on popular negative items and reduces the difficulty of unpopular negative items, which have a higher probability of being false negatives.

- To examine the effect of fine-grained hardness, we conduct experiments with four different hardness learning strategies, including AdvInfoNCE, InfoNCE, InfoNCE-Rand, and AdvInfoNCE-Reverse. AdvInfoNCE-Reverse refers to a strategy where hardness is learned by minimizing, rather than maximizing, the loss function. This inversion results in what we term 'reversed hardness', in contrast to the approach of our AdvInfoNCE method. InfoNCE-Rand denotes the assignment of uniformly random hardness for each negative item. We conduct 5-fold experiments with different random seeds for each strategy and report the mean value with standard error in Figure 4c. As the result shows, AdvInfoNCE yields consistent improvements over the other three different hardness strategies during the training phase. In contrast, the performance of AdvInfoNCE-Reverse drops rapidly as continuously training the reversed hardness. The sustained superior performance of AdvInfoNCE indicates that it effectively promotes the generalization ability of the CF model by automatically distinguishing false negatives and hard negatives.

- Figure 4b illustrates the uniform and align loss during the training phase following the initial warm-up epochs. As demonstrated, after the warm-up phase, both InfoNCE and InfoNCE-Rand exhibit a slight increase in align loss, while their uniform loss maintains a stable level. In contrast, AdvInfoNCE significantly improves uniformity at an acceptable cost of increasing align loss. On the other hand, employing reversed hardness (as in AdvInfoNCE-Reverse) appears to have a negative impact on representation uniformity. These findings underscore the importance of fine-grained hardness in AdvInfoNCE, suggesting that AdvInfoNCE learns more generalized representations.

D.3 Effect of the Adversarial Training Epochs on KuaiRec

In this section, we conduct experiments on KuaiRec, where AdvInfoNCE is trained for varying numbers of adversarial epochs. We plot performance metrics (Recall@20 in Figure 5a and NDCG@20 in Figure 5b) on the test set, which represents out-of-distribution data, throughout the training phase. The green stars mark the corresponding endpoint of adversarial training. As illustrated in Figure 5a and 5b, both Recall@20 and NDCG@20 show a proportional trend with the number of adversarial training epochs, up to a certain threshold. However, it is worth noting that in the extreme condition when the number of adversarial training epochs exceeds the threshold, performance on out-of-distribution sharply declines. This indicates a need to strike a balance when determining the appropriate number of adversarial training epochs for AdvInfoNCE.

Table 9: Training cost on Tencent (seconds per epoch/in total).

	+InfoNCE	+SGL	+NCL	+XSimGCL	+CCL	+BC loss	+Adap-τ	+AdvInfoNCE
MF	16.8 / 7,123	–	–	–	17.2 / 4,111	19.1 / 6,751	22.3 / 7,694	21.6 / 11,534
LightGCN	41.2 / 21,177	82.5 / 4,868	54.9 / 5,161	42.1 / 842	42.1 / 11,114	43.5 / 23,664	55.6 / 17,236	44.6 / 21,586

Algorithm 1 AdvInfoNCE

```
Input: observed interactions $\mathcal{O}^+$, unobserved interactions $\mathcal{O}^-$, learning rate of adversarial training $lr_{adv}$, maximum adversarial training epochs $E_{adv}$, adversarial training intervals $T_{adv}$, parameters of the CF model $\theta$, parameters of the hardness evaluation models $\theta_{adv}$, weighting parameter $K$
Output: $\theta$
Initialize: Initialize $\theta$ and $\theta_{adv}$, $e \leftarrow 1$, $e_{adv} \leftarrow 1$
repeat
    Freeze parameters of the hardness evaluation model $\theta_{adv}$
    Randomly sample $N$ negative items from $\mathcal{I}_u^-$ for each interaction within a batch
    Compute $s(u,i)$, $\delta_j^{(u,i)}$ with $\theta$ and $\theta_{adv}$, respectively
    Compute $\mathcal{L}_{\text{Adv}}(u,i) = -\log \frac{\exp(s(u,i))}{\exp(s(u,i)) + K\sum_{j=1}^{N}\exp(\delta_j^{(u,i)})\exp(s(u,j))}$
    Update $\theta$ by minimizing $\mathcal{L}_{\text{Adv}}(u,i)$
    if $e \bmod T_{adv} == 0 \quad \& \quad e_{adv} \leq E_{adv}$ then
        Freeze parameters of the CF model $\theta$
        Update $\theta_{adv}$ by maximizing $\mathcal{L}_{\text{Adv}}(u,i)$
        $e_{adv} \leftarrow e_{adv} + 1$
    end if
    $e \leftarrow e + 1$
until CF model converges
```

D.4 Hardness Learning Strategy

To accurately evaluate the hardness of each negative instance, we need to establish a mapping from unobserved user-item pairs to their corresponding hardness values, and this mapping mechanism can be diversified. Generally, the hardness learning strategy can be formulated as:

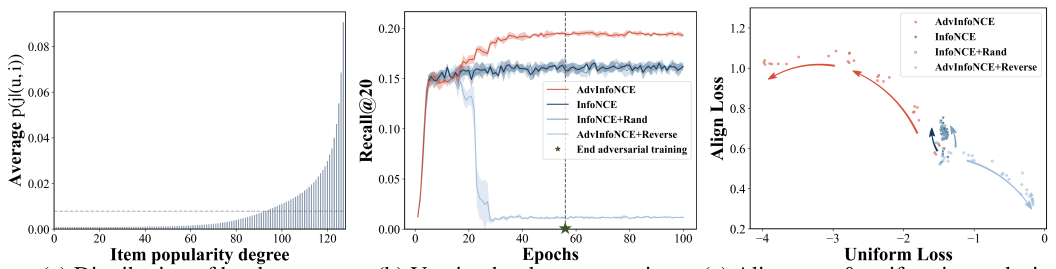
$$\delta_j^{(u,i)} \doteq \log(|\mathcal{N}_u| \cdot p(j|(u,i))) \tag{28}$$

$$\doteq \log\left(|\mathcal{N}_u| \cdot \frac{\exp\left(g_{\theta_{adv}}(u,j)\right)}{\sum_{k=1}^{|\mathcal{N}_u|}\exp(g_{\theta_{adv}}(u,k))}\right), \tag{29}$$

where $g_{\theta_{adv}}(u,j)$ is a raw hardness score function for the unobserved user-item pair (u,j), and $p(j|(u,i))$ is the probability of sampling the negative instance j, which is calculated by normalizing $g_{\theta_{adv}}(u,j)$. In this paper, we proposed two specific mapping methods: embedding-based (*i.e.,* AdvInfoNCE-embed) mapping and multilayer perceptron-based (*i.e.,* AdvInfoNCE-mlp) mapping. It should be noted that all the results reported in the main text of this paper are based on the implementation of the AdvInfoNCE-embed version.

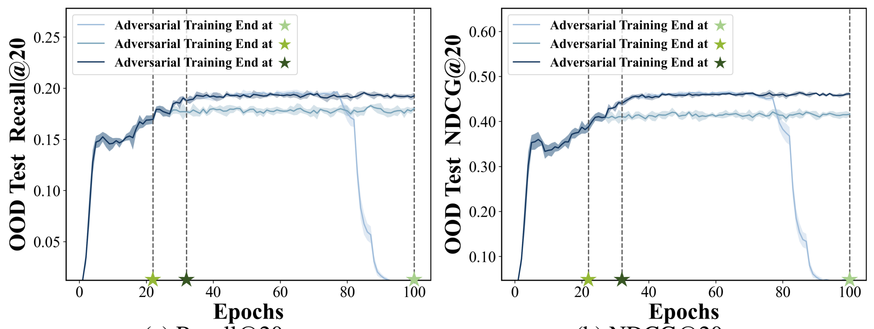
AdvInfoNCE-embed. The hardness computation process in AdvInfoNCE-embed follows a similar protocol as CF models. We directly map the index of users and items into its corresponding hardness embedding and calculate the hardness of each user-item pair through a score function. Specifically, this process involves a user hardness encoder $\psi_{\theta_{adv}}(\cdot) : \mathbb{U} \rightarrow \mathbb{R}^d$ and an item hardness encoder $\phi_{\theta_{adv}}(\cdot) : \mathbb{I} \rightarrow \mathbb{R}^d$, where θ_{adv} denotes all the trainable parameters of the hardness learning model. In our experiments, we adopt the same embedding dimension as the CF models for hardness learning. For the score function, we define $g_{\theta_{adv}}(u,j) = \psi_{\theta_{adv}}(u)^\top \cdot \phi_{\theta_{adv}}(j)$.

AdvInfoNCE-mlp. Unlike AdvInfoNCE-embed, AdvInfoNCE-mlp maps the embeddings of items and users from the CF model into another latent space with two multilayer perceptrons (MLPs) and calculates the hardness on this latent space. By respectively defining the MLPs for users and items as MLP_u and MLP_v, the score function for hardness calculation is defined as $g_{\theta_{adv}}(u,j) = \text{MLP}_\text{u}\left(\psi_\theta(u)\right)^\top \cdot \text{MLP}_\text{v}\left(\phi_\theta(j)\right)$. In our experiment, we simply employ one-layer MLPs and set the dimension of latent space as four. It's worth noting that this MLP-based implementation of



(a) Distribution of hardness (b) Varying hardness strategies (c) Alignment & uniformity analysis

Figure 4: Study of hardness. (4a) Illustration of hardness *i.e.,* the probability of negative sampling $(p\,(j|\,(u,i)))$ learned by AdvInfoNCE *w.r.t.* item popularity on KuaiRec. The dashed line represents the uniform distribution. (4b) Performance comparisons with varying hardness learning strategies on KuaiRec. (4c) The trajectories of align loss and uniform loss during training progress. Lower values indicate better performance. Arrows denote the losses' changing directions.



(a) Recall@20 (b) NDCG@20

Figure 5: The performance on KuaiRec with different numbers of adversarial training epochs. (5a) Performance comparisons *w.r.t.* Recall@20 on KuaiRec. (5b) Performance comparisons *w.r.t.* NDCG@20 on KuaiRec.

AdvInfoNCE may also be adaptable for handling out-of-distribution tasks in other fields, such as computer vision (CV) and natural language processing (NLP).

As reported in Table 10, both AdvInfoNCE-embed and AdvInfoNCE-mlp yield significant improvements over InfoNCE. Moreover, AdvInfoNCE-embed generally outperforms AdvInfoNCE-mlp.

D.5 The Intuitive Understanding of AdvInfoNCE

In this section, we aim to understand the mechanism of AdvInfoNCE intuitively, from the perspective of false negative identification.
Figure 6 illustrates the changes of the out-of-distribution performance and FN identification rate during the training process on Tencent. Here, the FN identification rate indicates the proportion of false negatives with negative δ_j. It can be observed that the out-of-distribution performance exhibits a rising trend along with the FN identification rate. Meanwhile, the out-of-distribution performance of InfoNCE remains relatively low. This indicates that AdvInfoNCE enhances the generalization ability of the CF model by identifying false negatives.

Figure 7 illustrates how AdvInfoNCE adjusts the scores and rankings of sampled negative items, by identifying the false negatives and true negatives. We retrieve the negative items sampled during training. If a sampled negative item appears in the test set, it is labeled as a false negative (FN); otherwise. In Figure 7a and 7b, the leftmost item represents a false negative, while the other two items on the right are negatives. The bar charts in blue and red depict the cosine similarity scores of sampled negative items measured by InfoNCE and AdvInfoNCE respectively. The rankings of sampled negative items are annotated above the bars. The line graph illustrates the hardness δ_j computed by AdvInfoNCE, measured on the right axis. It can be observed that when the hardness δ_j

Table 10: The performance comparison between AdvInfoNCE-embed and AdvInfoNCE-mlp over the LightGCN backbone.

	KuaiRec		Yahoo!R3		Coat		Tencent (γ = 200)		Tencent (γ = 10)		Tencent (γ = 2)	
	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG
InfoNCE	0.1800	0.4529	0.1475	0.0698	0.2689	0.1882	0.0540	0.0320	0.0332	0.0195	0.0242	0.0145
AdvInfoNCE-embed	0.1979	0.4697	0.1527	0.0718	0.2846	0.2026	0.0594	0.0356	0.0403	0.0243	0.0295	0.0180
Imp.% over InfoNCE	9.94%	3.71%	3.53%	2.87%	5.84%	7.65%	10.00%	11.25%	21.39%	24.62%	21.90%	24.14%
AdvInfoNCE-mlp	0.1851	0.4579	0.1545	0.0724	0.2843	0.2002	0.0567	0.0339	0.0364	0.0221	0.0260	0.0160
Imp.% over InfoNCE	2.83%	1.10%	4.75%	3.72%	5.73%	6.38%	5.00%	5.94%	9.64%	13.33%	7.44%	10.34%

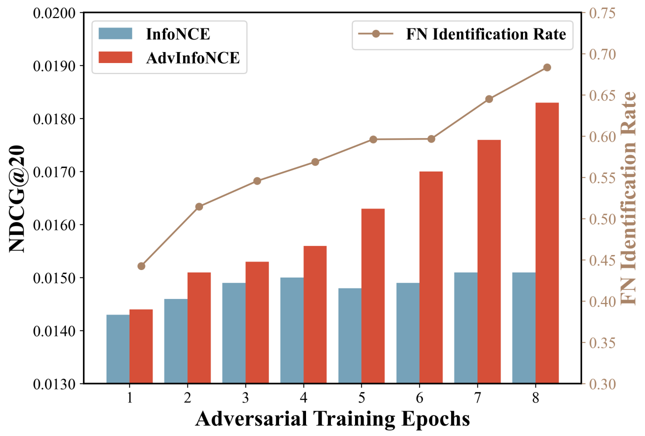
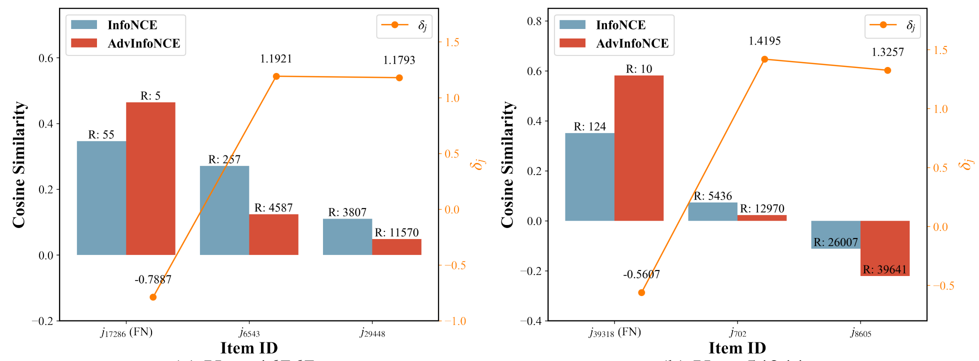


Figure 6: FN identification rate and NDCG@20 during training on Tencent, where FN identification rate indicates the proportion of false negatives (FN) with negative δ_j and NDCG@20 shows the out-of-distribution performance. As training proceeds, AdvInfoNCE' FN identification rate increases, capping at nearly 70%. This reveals AdvInfoNCE's capability to identify approximately 70% of false negatives in the test set. We attribute the superior recommendation performance of AdvInfoNCE over InfoNCE to this gradual identification.

is negative (*i.e.,* indicating that the item is identified as a false negative), the cosine similarity score improves. Conversely, if the hardness δ_j is positive, the score decreases.

E Hyperparameter Settings

For a fair comparison, we conduct all the experiments in PyTorch with a single Tesla V100-SXM3-32GB GPU and an Intel(R) Xeon(R) Gold 6248R CPU. We optimize all methods with the Adam optimizer and set the layer numbers of LigntGCN by default at 2, with the embedding size as 64 and the weighting parameter K as 64. We search for hyperparameters within the range provided by the corresponding references. For AdvInfoNCE, we search lr_{adv} in [1e-1, 1e-4], E_{adv} in [1, 30] and T_{adv} in $\{5, 10, 15, 20\}$. We adopt the early stop strategy that stops training if Recall@20 on the validation set does not increase for 20 successive evaluations. It's worth noting that AdvInfoNCE inherits the hyperparameter sensitivity property of adversarial learning, therefore it's necessary to choose proper hyperparameters for different datasets. We suggest selecting a suitable adversarial learning rate lr_{adv} first and then increasing the number of adversarial training epochs E_{adv} gradually until AdvInfoNCE reaches a relatively stable performance. We report the effect of changing the number of negative sampling in Table 13, where N is the number of negative sampling.



(a) User 46767 (b) User 54944

Figure 7: Case studies of refining the item ranking. With two randomly sampled users along with their sampled negative items, we subsequently retrieve their associated δ values, ranking positions, and cosine similarities. Here FN denotes false negative (*i.e.,* interactions unobserved during training but present in testing). The bar charts demonstrate the cosine similarity scores of these sampled negative items as gauged by both InfoNCE and AdvInfoNCE. Their rankings are annotated atop the bars. An accompanying line illustrates the hardness δ_j derived by AdvInfoNCE (measured on the right axis). Notably, when $\delta < 0$, AdvInfoNCE identifies and elevates an FN; conversely, for a potentially true negative, AdvInfoNCE leans towards a positive δ and declines its rank.

Table 11: Hyperparameters search spaces for baselines.

	Hyperparameter space

MF & LightGCN	lr $\sim$ {1e-5, 3e-5, 5e-5, 1e-4, 3e-4, 5e-4, 1e-3}, batch size $\sim$ {64, 128, 256, 512, 1024, 2048}
	No. negative samples $\sim$ {64, 128, 256, 512}

SSM	$\tau \sim$ [0.05, 3]
CCL	$w \sim$ {1, 2, 5, 10, 50, 100, 200}, $m \sim$ {0.2, 0.4, 0.6, 0.8, 1}
BC Loss	$\tau_1 \sim$ [0.05, 3], $\tau_2 \sim$ [0.05, 3]
Adap-τ	$warm\_up\_epochs \sim$ {10, 20, 50, 100}
SGL	$\tau \sim$ [0.05, 3], $\lambda_1 \sim$ {0.005, 0.01, 0.05, 0.1, 0.5, 1.0}, $\rho \sim$ {0, 0.1, 0.2, 0.3, 0.4, 0.5}
NCL	$\tau \sim$ [0.05, 3], $\lambda_1 \sim$ [1e-10, 1e-6], $\lambda_2 \sim$ [1e-10, 1e-6], $k \sim$ [5, 10000]
XSimGCL	$\tau \sim$ [0.05, 3], $\epsilon \sim$ {0.01, 0.05, 0.1, 0.2, 0.5, 1.0}, $\lambda \sim$ {0.005, 0.01, 0.05, 0.1, 0.5, 1.0}, $l* = 1$

Table 12: Model architectures and hyperparameters for AdvInfoNCE.

	Hyperparameters of AdvInfoNCE						
	lr_{adv}	E_{adv}	T_{adv}	τ	lr	batch size	No. negative samples
LightGCN							
Tencent	5e-5	7	5	0.09	1e-3	2048	128
KuaiRec	5e-5	12	5	2	3e-5	2048	128
Yahoo!R3	1e-4	13	5	0.28	5e-4	1024	64
Coat	1e-2	20	15	0.75	1e-3	1024	64
MF							
Tencent	5e-5	8	5	0.09	1e-3	2048	128
Yahoo!R3	1e-4	12	5	0.28	5e-4	1024	64
Coat	1e-2	18	15	0.75	1e-3	1024	64

Table 13: Varying number of negative sampling on Tencent

	γ = 200			γ = 10			γ = 2			Validation
N	HR	Recall	NDCG	HR	Recall	NDCG	HR	Recall	NDCG	NDCG
64	0.1513	0.0563	0.0333	0.1006	0.0373	0.0225	0.0708	0.0269	0.0164	0.0854
128	0.1600	0.0594	0.0356	0.1087	0.0403	0.0243	0.0774	0.0295	0.0180	0.0879
256	0.1642	0.0609	0.0367	0.1125	0.0419	0.0253	0.0815	0.0310	0.0189	0.0889