# Towards More Robust and Accurate Sequential Recommendation with Cascade-guided Adversarial Training

Juntao Tan[*]
Rutgers University
juntao.tan@rutgers.edu

Shelby Heinecke
Salesforce Research
shelby.heinecke@salesforce.com

Zhiwei Liu
Salesforce Research
zhiweiliu@salesforce.com

Yongjun Chen[†]
Apple Inc.
yongjun_chen@apple.com

Yongfeng Zhang
Rutgers University
yongfeng.zhang@rutgers.edu

Huan Wang
Salesforce Research
huan.wang@salesforce.com

## ABSTRACT

Sequential recommendation models, models that learn from chronological user-item interactions, outperform traditional recommendation models in many settings. Despite the success of sequential recommendation, their robustness has recently come into question. Two properties unique to the nature of sequential recommendation models may impair their robustness - the cascade effects induced during training and the model's tendency to rely too heavily on temporal information. To address these vulnerabilities, we propose Cascade-guided Adversarial training, a new adversarial training procedure that is specifically designed for sequential recommendation models. Our approach harnesses the intrinsic cascade effects present in sequential modeling to produce strategic adversarial perturbations to item embeddings during training. Experiments on training state-of-the-art sequential models on four public datasets from different domains show that our training approach produces superior model ranking accuracy and superior model robustness to real item replacement perturbations when compared to both standard model training and generic adversarial training.

## CCS CONCEPTS

**• Information systems → Recommender systems**; **• Theory of computation → Adversarial learning**.

## KEYWORDS

Robustness, Adversarial Training, Sequential Recommendation

**ACM Reference Format:**
Juntao Tan, Shelby Heinecke, Zhiwei Liu, Yongjun Chen, Yongfeng Zhang, and Huan Wang. 2018. Towards More Robust and Accurate Sequential Recommendation with Cascade-guided Adversarial Training. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX)*. ACM, New York, NY, USA, 11 pages. https://doi.org/XXXXXXX.XXXXXXX

[*]This project was completed while participating in an internship at Salesforce Research.
[†]This project was completed while working at Salesforce Research.

## 1 INTRODUCTION

Sequential recommender models learn dynamic user preferences and recommend next items by modeling past user behaviors in sequential order. Recently, deep learning based models have gained attention. These models learn user preferences by feeding the user histories into deep neural networks, such as RNN [13, 14, 42], CNN [36, 44], or transformer [15, 19, 34], and generally outperform the traditional sequential recommendation models [8, 39].

Despite their outstanding performance, recent works reveal their unique robustness issues. A small change at the end of the user sequence, sometimes even on only one item, can dramatically change model behavior [46, 47]. In real world applications, this kind of instability can lead to serious user dissatisfaction, e.g., sometimes a user misclicks an unwanted video, when they get back to the previous page, all the recommended videos are affected by this misclicking. In general, robustness is an important property for recommender systems as unrobust recommendation can decrease user trust and acceptance of the recommendation results [1, 20, 28]. Model accuracy must be prioritized alongside robustness. Prior works show that robustness and accuracy are sometimes correlated [28], hence training a more stable recommender system may also lead to more accurate predictions. Previous work in robustness and stability in recommender systems focus on model families such as matrix factorization [7, 12, 30, 37, 38, 41, 45]. Improving the robustness of complex models such as sequential recommendation models, while simultaneously improving accuracy, is still relatively unexplored.

Two properties unique to sequential recommendation models may hinder their robustness: (1) sequential recommendation models leverage temporal information to learn user preferences, but the model tends to **overuse** this temporal information - interactions that occur most recently in the user history influence the model recommendations the most, leading to over-sensitivity to these interactions [46, 47]; (2) intrinsic *cascade effects* exist in trained sequential models because of their time-aware training regime [27]. Cascade effects refers to the mechanism where changes in one user indirectly influence the recommendation for other users due to collaborative filtering [5, 27]. In sequential recommender models, item interactions in early timestamps generally induce larger cascade effects, since they affect the predictions on every subsequent interaction in each training epoch [27]. During training, their gradients are computed more frequently and are more diverse. When making predictions, the trained model may be more robust to perturbations occurring on the interactions with higher cascade values

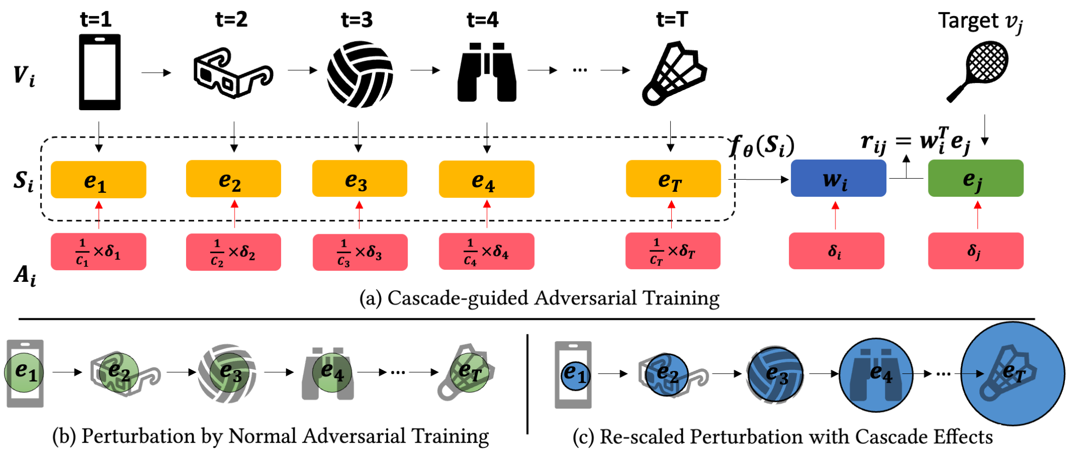

t=1
t=2
t=3
t=4
t=T
Target $v_j$
$V_i$
$S_i$
$A_i$
$e_1$
$e_2$
$e_3$
$e_4$
$e_T$
$f_\theta(S_i)$
$r_{ij} = w_i^T e_j$
$w_i$
$e_j$
$\frac{1}{c_1}\times\delta_1$
$\frac{1}{c_2}\times\delta_2$
$\frac{1}{c_3}\times\delta_3$
$\frac{1}{c_4}\times\delta_4$
$\frac{1}{c_T}\times\delta_T$
$\delta_i$
$\delta_j$
(a) Cascade-guided Adversarial Training
(b) Perturbation by Normal Adversarial Training
(c) Re-scaled Perturbation with Cascade Effects

**Figure 1: A toy example of applying adversarial training on sequential recommendation. (a) How adversarial perturbations are applied on the learned item and user embeddings. (b) Generic adversarial training applies adversarial perturbations of the same magnitude (green circles) to all item embeddings. (c) The Cascade-guided adversarial training method dynamically choose the magnitude of the perturbations (blue circles) according to the different cascade effects of each interaction in the user history.**

(i.e., the "early interactions") and less robust to perturbations on interactions with lower cascade values (i.e., the "later interactions"). The above two properties exactly contradict each other: during inference time, the most recent items in user history sequences may disproportionately influence predictions yet these recent items may be represented relatively infrequently during model training. The combination of these properties exacerbates the vulnerability to perturbations at the end of the user history sequences.

In this work, we alleviate the robustness issues described above by intervening into the training process. Since the interactions with lower cascade effects are trained less but commonly contribute more when making predictions, our basic idea is to increase the stability for such low-cascade interactions by adding more noise on them during training via adversarial training. Adversarial training has been shown to improve the robustness of a variety of deep learning models by adding adversarial perturbation on the input data [40]. Originally proposed for image classification, adversarial training produces image classification models that are more robust to adversarial attack on the input pixels [21]. In the sequential recommendation setting, we expect that by adding greater adversarial perturbations to the interactions with lower cascade effects, the trained model will be more robust to perturbations occurring among the most recent items in the user sequences when making predictions. Based on this, we propose a Cascade-guided Adversarial Training strategy, which is a re-normalized adversarial training method that is specifically designed for deep sequential recommendation by harnessing the intrinsic cascade effects for each interaction.

Our proposed method differs from prior work as follows: (1) Prior works develop adversarial training approaches for various recommendation model families such as matrix factorization, collaborative filtering, factorization machines, among others [6]. Adversarial training applied to deep sequential recommendation models, however, is mostly unexplored, and there are no works that leverage the unique properties of sequential recommender models as we do; (2) Prior works use adversarial training to improve generalization and/or improve robustness to adversarial perturbations on latent model embeddings. In contrast, we focus on improving robustness to adversarial perturbations on actual inputs, namely, item interaction sequences, since this is a more feasible type of perturbation that could occur in real-world systems.

In summary, our work makes the following contributions:

- We propose a novel adversarial training algorithm that uses the cascade effects to construct strategic adversarial examples during training.
- We conduct extensive experiments on four public datasets with two state-of-the-art sequential recommendation models. We show that, compared to standard model training and generic adversarial training, our approach produces more accurate recommendation models with superior model generalization and superior model robustness to realistic item replacement perturbations.

## 2 RELATED WORKS

In this section, we conduct a comprehensive review of prior studies on adversarial training and the robustness of recommender systems, and highlight the distinct characteristics of our proposed method in comparison to previous approaches.

### 2.1 Adversarial Training

Adversarial training emerged as a response to the seminal paper [35], where it was demonstrated that images with human-imperceptible perturbations, called adversarial examples, could be misclassified by powerful computer vision classifiers. Adversarial training is a framework for training models to be robust to such imperceptible perturbations in the data. See [40] for a general overview of adversarial training. Important developments in adversarial training include the Fast Gradient Sign Attack (FGSA) to generate adversarial examples [10] and the min-max robust optimization formulation of adversarial training [21]. In computer vision, adversarial training of image classifiers improves their robustness to adversarial examples [21] but, in some cases, adversarial training can harm generalization on unperturbed data [31].

In computer vision settings, adversarial examples model viable occurrences - an adversary could indeed directly manipulate the pixels of an image to fool the classifier. Adversarial training has recently been adapted to other fields, such as NLP and recommendation, but the adapted notions of adversarial examples are more abstract. For instance, in NLP, adversarial training is implemented via perturbations to word embeddings [23]. In recommendation settings, adversarial perturbation is also generally applied to user and item embeddings during training [12]. However, we argue that perturbing embeddings does not correspond to realistic perturbations in recommendation systems. In recommendation settings, a user (good or adversarial) usually does not have direct access to the recommendation model internals, such as the embeddings, but only has access to item interactions. Hence, perturbations to interactions caused by item misclicks are perhaps more representative than direct perturbations to user and item embeddings. In this work, we adopt the perspective that adversarial perturbations may occur on model externals, such as item interactions. To the best of our knowledge, adversarial training has yet to be explored as a method to enhance robustness against perturbations on model externals.

We also note that adversarial training approaches in the recommendation systems literature focus on model families such as matrix factorization, collaborative filtering, factorization machines, and auto-encoder models; see [6] and references therein. To our knowledge, only one prior work attempts to apply adversarial training to a sequential recommendation model, which is represented in our baseline methods [22]. There are no prior works that develop a general adversarial training for sequential recommendation based on their unique intrinsic properties as in our work.

## 2.2 Robustness of Recommendation

The definitions of a robust recommender system can be roughly classified into three categories. The first, and most widely accepted, definition of a robust recommender system is based on the model's ability to make accurate predictions in the presence of noisy data [9, 26, 28, 33]. These works suggest that ratings may not always reflect the user's true preferences because users are commonly careless and provide inaccurate data. The robustness of recommendation is measured as the change in accuracy as a function of the amount of noise present in the data. The second definition of robustness is based on the stability of the model's recommendations [1–3, 20]. Stability is generally measured by observing model predictions on unknown items after adding the algorithm's own predictions into the training dataset. A consistent model's predictions should not change dramatically in this setting. The third definition of robustness is defined from an attack and defense perspective [6, 46]. A third party may maliciously attack the recommendation models through injecting fake user profiles (i.e., shilling attacks). The goal of such attacks could be to promote certain items or to damage the system. In this scenario, the robustness of the system is defined as its resilience to harmful attacks. See [6] for a study on attack and defense approaches in recommendation. See [29] for additional notions of robustness to consider for recommendation systems.

The key goal of our work is to train robust sequential recommendation models with respect to the first definition of robustness, where our noise model is an adversarial item replacement scheme. We motivate the adversarial item replacement by first showing that replacing later items in interaction histories has a stronger negative impact to ranking accuracy than replacing items earlier in interaction histories. We then develop our novel adversarial training algorithm and show that it enhances resilience to this type of noise and it enhances generalization on unperturbed data.

## 3 PRELIMINARY

In this section, we briefly introduce the basic ideas of adversarial training and sequential recommendation.

## 3.1 Adversarial Training

Adversarial training is defined as a robust optimization problem with saddle point (min-max) formulation [21]. Consider a general classification problem with $d$-dimensional input data $x \in \mathbb{R}^d$ and corresponding label $y \in \mathbb{Z}$ under data distribution $\mathcal{D}$. For a classification model $f_\theta$, the training goal is to minimize the risk $\mathbb{E}_{x,y\sim\mathcal{D}}[L(f_\theta(x), y)]$. Adversarial training aims to find a perturbation $\delta$ with bounded norm $||\delta|| < \epsilon$ that maximizes the minimum risk with respect to $\theta$. This can be summarized as the following min-max equation:

$$\min_{\theta}\left(\max_{\delta,||\delta||<\epsilon} \mathbb{E}_{x,y\sim\mathcal{D}}[L(f_\theta(x+\delta), y)]\right) \tag{1}$$

Several methods have been proposed to find the optimal $\delta$ [4, 10, 21]. In this work, we invoke the most commonly used method that solves for $\delta$, fast gradient sign method (FGSM) [10]. FGSM simply generates adversarial perturbations by multiplying the sign of the gradient of the loss function by the maximal perturbation magnitude, $\epsilon$:

$$\delta = \epsilon \cdot sign\big(\nabla L(f_\theta(x+\delta), y)\big) \tag{2}$$

We note that the minimized adversarial training loss function is not necessarily the same as the general training loss function. Sometimes, adversarial training only minimizes the statistical distance between the original and new predicted probability distributions, regardless of the correctness of the original prediction (e.g., minimize KL-divergence). This is also referred to as virtual adversarial training [24].

## 3.2 Sequential Recommendation

Our work focuses on adversarial training for sequential recommendation systems. Sequential recommendation systems learn from the ordering of historical user-item interactions to predict the next user-item interaction. We formalize the key components as follows. Let $i \in [1, m]$ denote the index of a user and and $\mathcal{V} = \{v_1, v_2, \cdots, v_n\}$ denote the set of all possible items. Suppose user $i$ has a sequentially ordered interaction history of length $T$, $\mathcal{V}_i = \{v_i^t \mid t = 1, \cdots, T\}$. In practice, each user has different length of history. $T$ is a hyper parameter that decides the maximum length of user history, such that only the last $T$ interactions is considered when making predictions. A sequential recommendation model learns the embedding

| Symbol | Description |
|---|---|
| $m$ | The number of users |
| $n$ | The number of items |
| $\mathcal{V}$ | The set of items in a recommender system |
| $\mathcal{V}_i$ | The interaction history of user $i$ |
| $T$ | The maximum length of user history |
| $\boldsymbol{e_j}$ | The latent embedding of item $j$ |
| $\boldsymbol{e_i^t}$ | The latent embedding of the $t$th item in user $i$'s history |
| $\boldsymbol{E}$ | Embedding matrix for all items |
| $\boldsymbol{S_i}$ | The sequence of embeddings of items in $\mathcal{V}_i$ |
| $f$ | A Sequence embedding model |
| $\boldsymbol{w_i}$ | The predicted user embedding for user $i$ |
| $r_{i,j}$ | The predicted ranking acore between user $i$ and item $j$ |
| $\mathcal{N}^-$ | Negative samples when training BCE loss |
| $\boldsymbol{A_i}$ | The adversarial perturbations on sequence embedding $\boldsymbol{S_i}$ |
| $\boldsymbol{\delta}$ | An adversarial perturbation on a single embedding |
| $\boldsymbol{C}$ | Matrix of cascade effects |
| $L_B$ | BCE loss on clean data |
| $L_{\text{adv-1}}$ | Adversarial loss on the first hierarchy of sequential model |
| $L_{\text{adv-2}}$ | Adversarial loss on the second hierarchy of sequential model |
| $L$ | Total loss for cascade-guided adversarial training |

**Table 1: Summary of the notations.**

for each item $v_i$ denoted as $\boldsymbol{e_i}$. Together, these embeddings form the item embedding matrix, $\boldsymbol{E} \in \mathbb{R}^{n\times d}$. For each user $i$, we concatenate the sequence of embeddings of items in $\mathcal{V}_i$, denoted as $\boldsymbol{S_i} = [\boldsymbol{e_i^t} \mid t = 1, \cdots, T]$. The sequential models learn the user embedding, $\boldsymbol{w}_i$, as a function of the sequence embeddings,

$$\boldsymbol{w_i} = f(\boldsymbol{S_i}; \theta), \tag{3}$$

where $f$ denotes a sequence embedding model and $\theta$ denotes the model parameters. In sequential recommender systems, $f$ can be a Recurrent Neural Network (RNN), Convolutional Neural Network (CNN), an Attention mechanism, etc. After learning $\boldsymbol{w}_i$, for a target item $v_j$, the ranking score $r_{i,j}$ is predicted by

$$r_{i,j} = \boldsymbol{w_i^T} \boldsymbol{e_j}. \tag{4}$$

During model training, for each user $i$ with target item $v_j$ and a set of negative samples $\mathcal{N}^- \subset \mathcal{V} \setminus v_j$, we minimize the binary cross entropy (BCE) loss defined as:

$$L_B(i, j, \mathcal{N}^-; \theta, \boldsymbol{E}) = -\big(\log(\sigma(r_{i,j})) + \sum_{v_n \in \mathcal{N}^-} \log(1 - \sigma(r_{i,n})\big) \tag{5}$$

During training, following the same setting as in [15], we truncate the user sequence $\mathcal{V}_i$ according to each timestamp $t$. For each subsequence, item $v_i^t$ is treated as the target item and the ranking score is predicted by taking previous items as dynamic user history. Meanwhile, only one negative item $v_n$ is sampled for each $v_i^t$ in each sub-sequence. For simplicity, in the rest of the paper, we denote the ranking loss for item $v_j$ to user $i$ with negative sample $v_n$ as $L_{\text{B}}(i, j, n)$.

## 4 METHOD

We first introduce the proposed adversarial training algorithm that considers the cascade effects in sequential recommendation. Then, we propose an algorithm to effectively compute the cascade effects for each interaction in the user histories. The major notations used in the rest of the paper are summarized in Table 1.

### 4.1 Cascade-guided Adversarial Training

Since recommender systems take discrete input data (i.e., user/item IDs), adversarial perturbations are commonly applied on the latent embeddings. As shown in Figure 1 (a), a deep sequential recommendation model is a hierarchy consisting of two levels: (1) a non-linear deep neural network that takes the sequence embeddings of the user's history $\boldsymbol{S_i}$ as input and outputs the user embedding $\boldsymbol{w_i}$; (2) a linear model that predicts the final ranking score for user $i$ on target item $v_j$ by linearly multiplying their embeddings as $\boldsymbol{w_i^T e_j}$. We apply adversarial training on both levels separately.

For the first level, the adversarial training objective is described as follows. When perturbing the item embeddings in the history sequence within a certain small magnitude, even in the worst case, the learned user embedding shouldn't be too different from the original prediction. Suppose for user $i$ with sequence embeddings $\boldsymbol{S_i}$, we denote the adversarial perturbations on the sequence embeddings as $\boldsymbol{A_i} = [\boldsymbol{\delta_i^t} \mid t = 1, \cdots, T]$, where $\boldsymbol{\delta_i^t}$ is the perturbation applied on the corresponding item embedding $\boldsymbol{e_i^t}$. We mathematically formulate this objective by an adversarial loss function:

$$\begin{aligned} &L_{\text{adv-1}}(i, \boldsymbol{A_i}) = ||\boldsymbol{\hat{w}_i} - \boldsymbol{w_i}||_2 \\ &\text{where } \boldsymbol{w_i} = f(\boldsymbol{S_i}; \theta), \quad \boldsymbol{\hat{w}_i} = f\big(\boldsymbol{S_i} + \frac{1}{\boldsymbol{C_i}} \odot \boldsymbol{A}_i; \theta\big) \end{aligned} \tag{6}$$

In Eq (6), the vector $\boldsymbol{C_i} \in [1, +\infty)^T$ denotes the cascade effects of each interaction in user $i$'s history. Higher values of $\boldsymbol{C_i}$ denote higher cascade effects. The way to calculate these cascade effects will be introduced in the next section. The factor $\frac{1}{\boldsymbol{C_i}}$ plays a crucial rule in our method: it re-scales the adversarial perturbations so that interactions with smaller cascade effects will receive a larger adversarial perturbation. During sequential recommendation model training, interactions with larger cascade effects are used more often in training than interactions with smaller cascade effects [27], hence, the latter can be more vulnerable and unstable (justified later in Section 5.6). By applying larger perturbations on the interactions with lower cascade effects, we obtain a model that is more equally robust across all sequence embeddings.

To approximate the worst case adversarial perturbation $\boldsymbol{A_i}$, we apply the FGSM attack introduced in Eq.(2):

$$\mathcal{A}_i = \epsilon \frac{\boldsymbol{g}}{||\boldsymbol{g}||_2} \quad \text{where } \boldsymbol{g} = \frac{\partial L_{\text{adv-1}}(i, \boldsymbol{A_i})}{\partial S_i} \tag{7}$$

We note that $\frac{\boldsymbol{g}}{||\boldsymbol{g}||_2}$ is the sign of the direction of the applied perturbation and $\epsilon$ is a human defined parameter to determine the general magnitude of the adversarial attack. The specific magnitude of the perturbation on each interaction will be re-scaled by their cascade effects, which is one of the key features of our method.

The adversarial training on the first level guarantees the aggregated user embedding $\boldsymbol{w_i}$ is robust to small perturbations on the user history. On the second level, we apply adversarial training on the learned user embedding and the target item embeddings. This assures that when small changes are applied to the user and target item embeddings, the model is still able to generate highly accurate recommendation results. We use BCE loss as the second

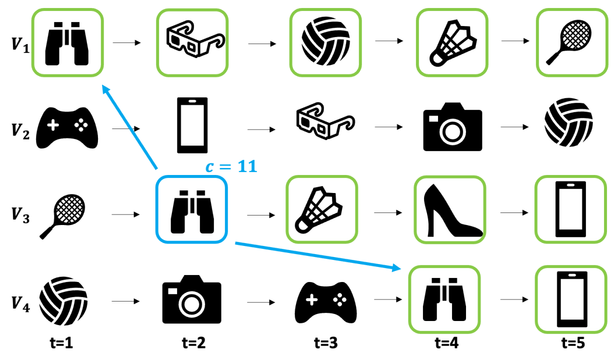


**Figure 2: An example of calculating cascade effects. The item with blue bounding box has cascade effects on the 10 items with green bounding boxes plus itself. Its cascade value is 11.**

adversarial training loss, which is exactly the same loss function used to train the base recommendation model. Suppose $\boldsymbol{\delta_i}$, $\boldsymbol{\delta_j}$, $\boldsymbol{\delta_n}$ are the adversarial perturbations applied on the embeddings of the user $i$, target item $v_j$, and negative sampled item $v_n$, respectively. The second adversarial training loss is defined as:

$$L_{\text{adv-2}}(i, j, n, \boldsymbol{\delta_i}, \boldsymbol{\delta_j}, \boldsymbol{\delta_n}) = -\big(\log(\sigma(\hat{r}_{i,j})) + \log(1 - \sigma(\hat{r}_{i,n})\big) \quad \text{where } \hat{r}_{i,j} = (\boldsymbol{w_i^T} + \boldsymbol{\delta_i})(\boldsymbol{e_j} + \boldsymbol{\delta_j}), \quad \hat{r}_{i,n} = (\boldsymbol{w_i^T} + \boldsymbol{\delta_i})(\boldsymbol{e_n} + \boldsymbol{\delta_n}) \tag{8}$$

Here $\hat{r}_{i,k}$ is the predicted ranking score for any user $i$ and item $k$ after perturbation. Similarly, we approximately generate the worst case $\boldsymbol{\delta_i}$, $\boldsymbol{\delta_j}$, and $\boldsymbol{\delta_n}$ within maximum magnitude $\epsilon$ by:

$$\begin{aligned} \boldsymbol{\delta_i} &= \epsilon \frac{\boldsymbol{h_i}}{||\boldsymbol{h_i}||_2} \quad \text{where } \boldsymbol{h_i} = \frac{\partial L_{\text{adv-2}}(i, j, n, \boldsymbol{\delta_i}, \boldsymbol{\delta_j}, \boldsymbol{\delta_n})}{\partial \boldsymbol{e_i}} \\ \boldsymbol{\delta_j} &= \epsilon \frac{\boldsymbol{h_j}}{||\boldsymbol{h_j}||_2} \quad \text{where } \boldsymbol{h_j} = \frac{\partial L_{\text{adv-2}}(i, j, n, \boldsymbol{\delta_i}, \boldsymbol{\delta_j}, \boldsymbol{\delta_n})}{\partial \boldsymbol{e_j}} \\ \boldsymbol{\delta_n} &= \epsilon \frac{\boldsymbol{h_n}}{||\boldsymbol{h_n}||_2} \quad \text{where } \boldsymbol{h_n} = \frac{\partial L_{\text{adv-2}}(i, j, n, \boldsymbol{\delta_i}, \boldsymbol{\delta_j}, \boldsymbol{\delta_n})}{\partial \boldsymbol{e_n}} \end{aligned} \tag{9}$$

The two adversarial training objectives synergistically improve the robustness of the trained model with respect to all the components. Finally, we optimize the model parameters by minimizing the sum of original BCE loss and the two adversarial training losses:

$$\min_{\theta, E} L = L_{\text{B}}(i, j, n) + \lambda_1 L_{\text{adv-1}}(i, \boldsymbol{A_i}) + \lambda_2 L_{\text{adv-2}}(i, j, n, \boldsymbol{\delta_i}, \boldsymbol{\delta_j}, \boldsymbol{\delta_n}) \tag{10}$$

It's worth noting that since the cascade effects only affect the model after the general training process, the adversarial training should be applied after the base recommendation has converged. In the adversarial training phase, by minimizing Eq. (10), we expect to learn better item embeddings $\boldsymbol{E}$ and the model parameters $\theta$ such that the model is more accurate and robust.

## 4.2 Cascade Effects Calculation

In this section, we will introduce how we calculate the cascade effect matrix $C \in [0, 1]^{m \times T}$ for all the interactions in the user histories, in a time efficient manner.

When training a sequential recommendation model, each user-item interaction produces cascade effects. For a given user-item interaction, two types of interactions receive its cascade effects: (1) all interactions following the given interaction, within the same user history sequence; (2) all interactions with the same item occurring in different user history sequences within the same training batch. This is illustrated in Figure 2.

Based on the above observation, for an item $v_t^i$ for which user $i$ interacted at timestamp $t$, we define the cascade effect $C(i, t)$ as:

$$C(i, t) = 1 + T - t + \frac{b}{m} \sum_{k \in \mathcal{U}, k \neq i} \sum_{l \leq T} (1 + T - l) \mathbb{I}(v_i^t, v_k^l) \tag{11}$$

$$\mathbb{I}(v_i^t, v_k^l) = \begin{cases} 1, & \text{if } v_i^t = v_k^l \\ 0, & \text{otherwise} \end{cases} \tag{12}$$

Here, $b$ is the batch size during training and $\frac{b}{m}$ approximates the probability of two user sequences appearing in the same training batch. After calculating $\boldsymbol{C}$, its inverse will be a real number in (0, 1], and this will be used to re-normalize the magnitudes of adversarial perturbations.

We note that in Eq. (11), $1 + T - t$ calculates the cascade effects that directly come from the temporal information in the **same** user history, i.e., the inverse of timestamp. The accumulative term calculates the cascade effects that comes from the same item in **different** user sequences. In Section 5.5, we do ablation study to show how much of each type of cascade effects contributes to the final performance.

In addition to Eq. (11), we propose Alg. 1, an efficient algorithm to compute the cascade scores for all ineractions in the user history sequences.

**Algorithm 1** Compute_Cascade_Effects

**Input:** user set $\mathcal{U}$, maximum sequence length $T$, user history sequences $\{\mathcal{V}_1, \mathcal{V}_2, \cdots \mathcal{V}_m\}$
**Output:** $C \in [0, 1]^{m \times T}$: The cascade effect of each interaction in each user's history (in last T timestamps).

```
1:  initialize $C$ with all 0, initialize $D \in \mathbb{R}^n$ with all 0
2:  //compute total cascade scores for each item
3:  for $t \in [T, T-1, ..., 1]$ do
4:      for $i \in \mathcal{U}$ do
5:          $D_{\mathcal{V}_i^t} \leftarrow (D_{\mathcal{V}_i^t} + 1 + T - t)$
6:      end for
7:  end for
8:  // fill cascade matrix $C$
9:  for $i \in \mathcal{U}$ do
10:     for $t \in [T, T-1, ..., 1]$ do
11:         $C_i^t = 1 + T - t + \frac{b}{m} D_{\mathcal{V}_i^t}$
12:     end for
13: end for
14: min-max normalize $C$
```

## 5 EXPERIMENTS

We conduct experiments to evaluate the generalization and robustness of deep sequential models trained under our proposed method. Specifically, we consider the following questions:

- **EXP1:** Does the proposed method improve the ranking accuracy of deep sequential recommendation models? How does it compare to normal adversarial training?

**Table 2: Statistics of the datasets.**

| Dataset | #User | #Item | #Interaction | Density |
|---|---|---|---|---|
| ML-1M | 6,040 | 3,416 | 987,540 | 4.786% |
| Beauty | 22,363 | 10,121 | 198,502 | 0.088% |
| Video | 24,303 | 10,672 | 231,780 | 0.089% |
| Clothing | 39,387 | 23,033 | 278,677 | 0.031% |

- **EXP2:** Does the proposed method improve the robustness of the trained models? If so, which aspect of robustness does it improve, and how much is the improvement?

In Section 5.1 and Section 5.2, we will first introduce the datasets and base recommendation models used in the experiments. Then, in Section 5.3 and Section 5.4, we will introduce the implementation details for training the models and the metrics used for evaluating the trained models. In Section 5.5, we will discuss the experiments on ranking accuracy. In Section 5.6, we will discuss the experiments on robustness. At last, in Section 5.7, we analyze the time efficiency of the proposed adversarial training algorithm.

## 5.1 Datasets

We conduct experiments on four public datasets from completely different domains with diverse densities:

- MovieLens-1M[1] [11]: A commonly used dataset in recommendation that consists of users' ratings on movies. It has 1 million ratings from over 6,000 users on over 3,000 movies.
- Amazon-Video, Amazon-Beauty & Amazon-Clothing [2] [25]: Amazon review datasets contain user ratings on products in the Amazon e-commerce system. From 29 available datasets in different product categories, we choose Video, Beauty, and Clothing for our experiments. Since these three datasets are very sparse, we filter out the users and items with less than 5 interactions. We follow previous works to retrieve the sequential order from the timestamp information [15, 32].

Table 2 shows the statistics of the datasets after pre-processing.

## 5.2 Base Models

Since RNN and transformer are the most common structures been used in deep sequential models [16], we choose two of the most representative recommendation models from each of the above categories as the base models:

- GRU4Rec[14]: GRU4Rec utilizes RNNs to learn user preferences based on their history sequences.
- SASRec[15]: SASRec relies on attention mechanisms that can dynamically learn the attention weights on each interaction in the user sequences.

## 5.3 Implementation Details

First, we implement exactly the same architectures for the two base models as described in their original papers. We use a single layer of GRU units in the GRU4Rec model and 2 self-attention blocks in the SASRec model. The hidden size is set to 100 for both models. When computing user embeddings, the maximum sequence length $T$ is set to 200 for MovieLens-1M and 50 for Video, Beauty and Clothing datasets according to their different densities.

We use same training strategy for training all the models on all the datasets: We first train the base models for 500 epochs to ensure their convergence, then we apply Adversarial Training (generic or our method) on the trained models for further 100 epochs. For both of the training phases, we use Adam optimizer[17] with 0.001 learning rate. The batch size is 128. We use 0.2 dropout rate and $1 \times 10^{-5}$ $L_2$ norm to prevent overfitting. We follow a leave-one-out strategy to split training and test data, which is commonly used in sequential recommendation.

For the hyper-parameters, the magnitude $\epsilon$ is always set to 10 for all the datasets. The ablation study on different $\epsilon$ values can be found in Section 5.5. For parameters $\lambda_1$ and $\lambda_2$ in Eq. (10) are always 1 such that the three loss functions equally contribute to the training. We note that there is no hyper-parameter used to compute the cascade effects, which means our proposed adversarial training method can be easily applied on new datasets and models without additional tuning effort.

## 5.4 Evaluating Metrics

We use standard evaluation metrics, Normalized Discounted Cumulative Gain (NDCG) and Hit Ratio (HT), to evaluate the ranking performances of the recommendation models in this paper. Noted that negative sampling is not used in the evaluation. Recent study shows that negative sampling causes inconsistency and leads to biased comparisons, and suggests that sampling should be avoided for metric calculation [18]. Instead, we rank over all items except the ones already in the user histories and retrieve the top-10 ranking list. This evaluation method is used in more and more state-of-the-arts [36, 43].

## 5.5 EXP1: Improvement on Ranking Accuracy

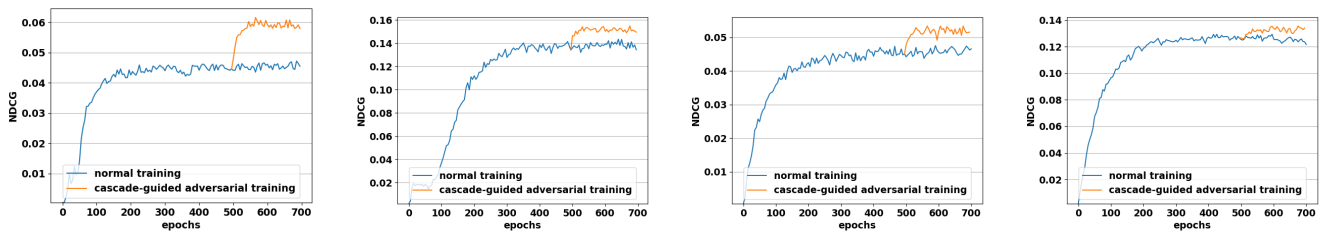
In the first set of experiments, we evaluate the extent to which our proposed adversarial training method can improve the ranking accuracy in comparison to baseline adversarial training methods. We consider three baseline adversarial training approaches:

- adv_linear[12]: An adversarial training method designed for MF-based recommender systems, which only applies the adversarial perturbation on the target user and item embeddings. In the sequential recommendation setting, this corresponds to applying perturbations only at the second level of the sequential recommendation model hierarchy, on $\boldsymbol{w_i}$ and $\boldsymbol{e_j}$.
- adv_sequence[22, 23]: [23] is an adversarial training method originally applied on an LSTM model for text classification. Later, [22] applies it on sequential recommendation. Though they are also applied on sequential models, these methods only add adversarial perturbations on the sequence embeddings.
- adv_global: A self-defined baseline. We use the same two-leveled adversarial training strategy as our proposed method but without cascade-guided re-normalization. Comparing to this baseline can be also seen as an ablation study of how

[1]https://files.grouplens.org/datasets/movielens
[2]https://nijianmo.github.io/amazon

**Table 3: Improvement of the ranking accuracy by applying different adversarial training methods on the base models.**

| | | MovieLens-1M | | | | Beauty | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | NDCG@10 | vs. Base | HT@10 | vs. Base | NDCG@10 | vs. Base | HT@10 | vs. Base |
| SASRec | base | 0.1359 | - | 0.2528 | - | 0.0264 | - | 0.0537 | - |
| | adv_linear[12] | 0.1508 | 10.96% | 0.2795 | 10.56% | 0.0315 | 19.32% | 0.0626 | 16.57% |
| | adv_seq[22, 23] | 0.1440 | 5.96% | 0.2680 | 6.01% | 0.0304 | 15.15% | 0.0597 | 11.17% |
| | adv_global | 0.1519 | 11.77% | 0.2808 | 11.08% | 0.0313 | 18.56% | 0.0622 | 15.83% |
| | **adv_cas** | **0.1546** | **13.76%** | **0.2831** | **11.99%** | **0.0320** | **21.21%** | **0.0630** | **17.32%** |
| GRU4Rec | base | 0.1255 | - | 0.2419 | - | 0.0228 | - | 0.0460 | - |
| | adv_linear[12] | 0.1351 | 7.65% | 0.2581 | 6.70% | 0.0288 | 26.32% | 0.0560 | 21.74% |
| | adv_seq[22, 23] | 0.1308 | 4.22% | 0.2520 | 4.18% | 0.0252 | 10.53% | 0.0493 | 7.17% |
| | adv_global | 0.1345 | 7.17% | 0.2543 | 5.13% | 0.0285 | 25.00% | 0.0550 | 19.57% |
| | **adv_cas** | **0.1360** | **8.37%** | **0.2588** | **6.99%** | **0.0298** | **30.70%** | **0.0566** | **23.04%** |
| | | Video | | | | Clothing | | | |
| | | NDCG@10 | vs. Base | HT@10 | vs. Base | NDCG@10 | vs. Base | HT@10 | vs. Base |
| SASRec | base | 0.0441 | - | 0.0875 | - | 0.0088 | - | 0.0184 | - |
| | adv_linear[12] | 0.0553 | 25.40% | 0.1059 | 21.03% | 0.0074 | -15.91% | 0.0154 | -16.30% |
| | adv_seq[22, 23] | 0.0519 | 17.69% | 0.1001 | 14.40% | 0.0106 | 20.45% | 0.0213 | 15.76% |
| | adv_global | 0.0557 | 26.30% | 0.1068 | 22.06% | 0.0103 | 17.05% | 0.0216 | 17.39% |
| | **adv_cas** | **0.0606** | **37.41%** | **0.1162** | **32.80%** | **0.0107** | **21.59%** | **0.0219** | **19.02%** |
| GRU4Rec | base | 0.0442 | - | 0.0872 | - | 0.0072 | - | 0.0150 | - |
| | adv_linear[12] | 0.0500 | 13.12% | 0.0962 | 10.32% | 0.0070 | -2.78% | 0.0154 | 2.67% |
| | adv_seq[22, 23] | 0.0456 | 3.17% | 0.0888 | 1.49% | 0.0074 | 2.78% | 0.0152 | 1.33% |
| | adv_global | 0.0496 | 12.22% | 0.0966 | 10.78% | 0.0071 | -1.39% | 0.0146 | -2.67% |
| | **adv_cas** | **0.0520** | **17.65%** | **0.0993** | **13.88%** | **0.0083** | **15.28%** | **0.0168** | **12.00%** |



(a) Training SASRec on Video. (b) Training SASRec on MovieLens-1M. (c) Training GRU4Rec on Video. (d) Training GRU4Rec on MovieLens-1M.

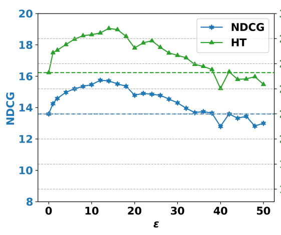
**Figure 3: NDCG vs. the number of training epochs. After the normal training process converges, the proposed adversarial training method can further improve accuracy of the models.**

**Table 4: How different components of cascade effects contribute to the final improvement. adv_cas_1 only considers the cascade effects from the same user history sequence. adv_cas_2 only considers the cascade effects from different user history sequences.**

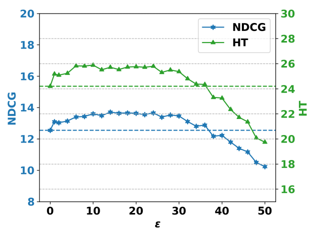
| Cascade type | MovieLens-1M | | Video | | Beauty | | Clothing | |
|---|---|---|---|---|---|---|---|---|
| | NDCG@10 | HT@10 | NDCG@10 | HT@10 | NDCG@10 | HT@10 | NDCG@10 | HT@10 |
| base | 0.1359 | 0.2528 | 0.0441 | 0.0875 | 0.0264 | 0.0537 | 0.0088 | 0.0184 |
| adv_cas_1 | 0.1523 | 0.2738 | 0.0583 | 0.1111 | 0.0309 | 0.0614 | 0.0106 | 0.0211 |
| adv_cas_2 | 0.1512 | 0.2704 | 0.0562 | 0.1101 | 0.0311 | 0.0614 | 0.0101 | 0.0204 |
| adv_cas | **0.1546** | **0.2831** | **0.0606** | **0.1162** | **0.0320** | **0.0630** | **0.0107** | **0.0219** |

much the cascade values contribute to the improvement of accuracy.

We note that for fair comparison, when implementing our method, we always re-normalize the average cascade values to 1 so that the

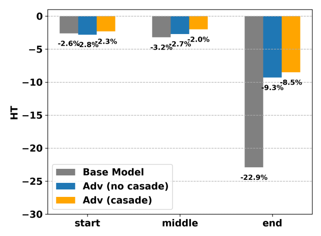


(a) SASRec on MovieLens-1M

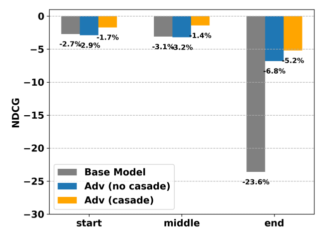


(b) GRU4Rec on MovieLens-1M

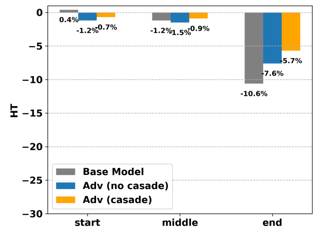
**Figure 4: Influence of $\epsilon$**



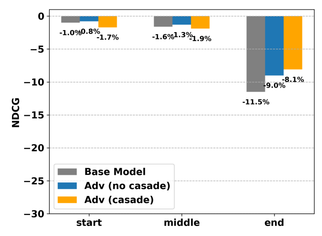
(a) HT@10: SASRec on Video dataset



(b) NDCG@10: SASRec on Video dataset



(c) HT@10: GRU4Rec on Video dataset



(d) NDCG@10: GRU4rec on Video dataset

**Figure 5: Drop of accuracy by attacking the the first, middle, and the last items in the user sequences respectively. The y-axis depicts negative values, with larger bars indicating larger decreases in accuracy of the recommendation models.**

total magnitudes of added perturbations are the same for all the baselines. The experiments results are shown in Table 3. First, we observe that compared to the pre-trained base models (no adversarial training), applying any of the adversarial training methods can generally further improve the ranking accuracy. This means that while adversarial training improves the models robustness on the adversarial examples, it also improve the models generalization on the clean data [23]. Second, our proposed Cascade-Guided Adversarial Training method consistently outperforms all the other baselines on all the datasets. On average, it improves the ranking accuracy of the base model by 15.32% for SASRec and 15.99% for GRU4Rec. Compared to the second best baselines, it shows 18.16% **more improvement** of the base model for SASRec and 168.94% for GRU4Rec. Third, the improvements on the sparser datasets (i.e., Video, Beauty, and Clothing) are more significant than the other dataset. This is expected since when training complicated deep neural networks on sparse datasets, the trained models are more easily to overfit on the data and less stable. It's worth noting that all the other adversarial training methods fail to improve GRU4Rec on Clothing dataset possibly due to it's extreme sparsity, but the cascade-guided adversarial training method still performs well.

We illustrate the learning curves of the two training phases in Figure 3. As shown in the figures, the base models converge in the first 500 epochs of pre-training. Training for more epochs can hardly benefits the trained models. However, when applying the Cascade-guided Adversarial Training, the ranking accuracy can be quickly improved further, especially in the next 100 steps.

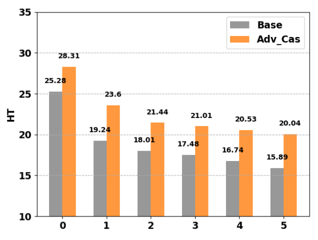
We also consider a related question: what should be the magnitude of the adversarial perturbations? Too little perturbation may not sufficiently improve the models' robustness. On the contrary, if the magnitudes of the perturbations are too large, the model may not learn any useful information. The magnitude of perturbations is controlled by the hyper-parameter $\epsilon$ in Eq. (7) and Eq. (9). We perform ablation studies on $\epsilon$, by setting its value from 0.1 to 50 as shown in Figure 4. The first observation is our proposed adversarial training algorithm is effective for a range of $\epsilon$ values. Even a very small number of $\epsilon$ (e.g., 1) can significantly improve the models' performance. The method shows the best performance when $\epsilon \in [10, 20]$ for the both base recommendation models. When $\epsilon$ is larger than 30, the models' accuracy start to drop, suggesting that the adversarial perturbations are too large, and useful information is lost during training.

As in Eq.(11), each interaction has two types of cascade effects on other interactions (i.e., cascade effects on direct later interactions and cascade effects on the interactions from other sequences). Additionally, we do ablation study about how each component of cascade effects contribute to the final performance. As shown in Table 4, only consider single type of cascade effects can still improve the trained models. However, the adversarial training method shows the best performance when considering both cascade effects.

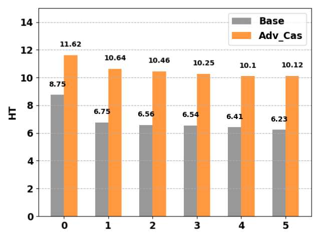
## 5.6 EXP2: Improvement on Robustness

The performance of recommendation models in the presence of noise is a key concern for measuring the robustness of a model [28]. Our proposed Cascade-guided Adversarial Training method improves the models' robustness to small perturbations on the users history, especially at the end of the user history sequences. It is worth noting that all the perturbations in the evaluation are based on real item replacement instead of adding noises on the item embeddings, hence, simply increasing the norm of the learned item embeddings can not lead to trivial solutions.

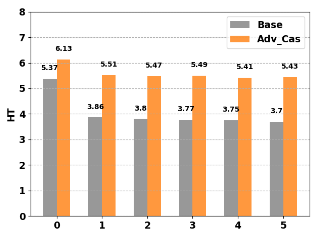
When replacing the items in the user sequences, we assume that the changed items are close to the original items in the embedding space, such that the perturbation is small. This is because if the user history change dramatically, a good recommender system is expected to provide different recommendations. Instead, replacing a similar item can be seen as a a mimic of the "natural" noise in data, such as misclicks. In the experiments, we always generate the worst case small replacement by first applying gradient-based attack on the embeddings of the items to be replaced, then matching with the closest items in the embedding space along the gradient direction.



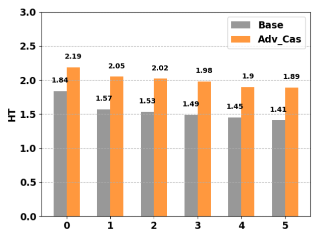
(a) SASRec on MovieLen-1M.



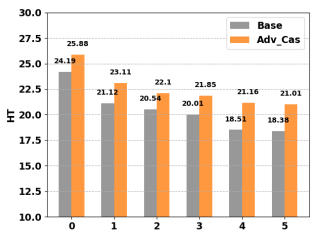
(b) SASRec on Video.



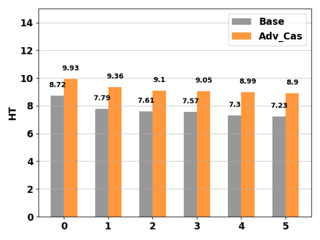
(c) SASRec on Beauty.



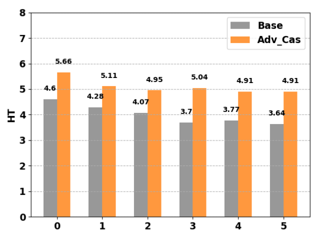
(d) SASRec on Clothing.



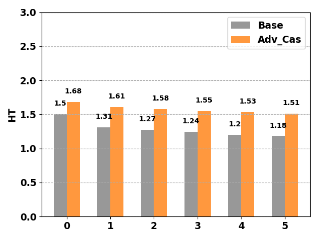
(e) GRU4Rec on MovieLens-1M.



(f) GRU4Rec on Video.



(g) GRU4Rec on Beauty.



(h) GRU4Rec on Clothing.

**Figure 6: Decreases of model accuracy by replacing K items at the end of user sequences according to HT@10.**

The intuition of our proposed method is that the trained sequential recommendation models are not equally robust along the entire user history; items toward the end of the user history are more vulnerable since adding noise there impacts performance more than adding noise to the beginning of the user history. We first do experiments to justify this hypothesis. We replace the first, middle, and the last items in the user sequences respectively, and measure the drop of the accuracy. As show in Figure 5, we test the two base models on the Video dataset. The results show that by replacing the first and the middle item of the user sequences, the accuracy of the prediction is only reduced by $0\% \sim 3\%$. However, by replacing the last item at the end of the sequences, the ranking accuracy drops significantly (i.e., 23.22% for NDCG and 11.04% for Hit ratio on average). This demonstrates the vulnerability of the base sequential recommendation models at the end of the sequences. The results show our Cascade-guided Adversarial Training successfully improves the models robustness at the end of the sequences, more than the original base models or the models trained with normal adversarial training methods.

Next, we evaluate model robustness by replacing the last $K$ items in user sequences. We choose $K$ from 1 to 5 to show the trend of decreasing model accuracy. We perform these experiments for both base models on all the four datasets, the results are illustrated in Figure 6. The results clearly show that the models trained with our Cascade-guided Adversarial Training consistently outperform the original models. Meanwhile, the decreasing accuracy is generally slower in most cases except for GRU4Rec on MovieLens-1M dataset, which shows similar percentage of decrease. The results show that the proposed training method indeed improves the model robustness for small substitutions. Note that for the Video, Beauty, and Clothing datasets, even by changing all the last 5 items in the original user sequences, the ranking accuracy is still higher than the normally trained models on clean data. This shows that our proposed method is significantly effective on sparse datasets.

## 5.7 Efficiency Analysis

The proposed adversarial training strategy requires extra training time. It comes from 1) the 100 extra training steps, 2) in each adversarial training step, the extra time required to compute the perturbations based on FGSM and 3) computing the cascade effects for each interaction (Alg. 1).

Among them, the impact of computing the cascade matrix can be omitted because of its linear time complexity of $O(mT)$. We test the running time of training SASRec on ML-1m dataset with a single Nvidia A100 GPU. The 100 adversarial learning steps take 8.3 minutes. Compared to the 10.6 minutes cost in the 500 steps of normal training stage, the cascade-guided adversarial training algorithm add 78% of the training time. However, given the significant improvements in ranking accuracy and robustness, we consider the extra time to be perfectly acceptable.

## 6 CONCLUSION

In this work, we introduce Cascade-guided Adversarial Training, a novel adversarial training algorithm designed for sequential recommender systems. This is the first adversarial training algorithm that considers the intrinsic properties of sequential recommendation models. Using our approach, we show that the trained sequential recommendation models are more robust and accurate. Considering the small number of additional training epochs and the simplicity of the additional parameters involved in the proposed training process, the proposed method is very practical.

Adversarial training in recommendation has a large potential, especially when customizing it for specific purposes. For instance, in this paper, the proposed method alleviates the unique robustness issues for sequential recommendation models. It could be designed differently for other objectives, such as de-biasing, improving the model generalization, etc. We expect adversarial training to be explored more in the future.

## REFERENCES

[1] Gediminas Adomavicius and Jingjing Zhang. 2010. On the stability of recommendation algorithms. In *Proceedings of the fourth ACM conference on Recommender systems*. 47–54.

[2] Gediminas Adomavicius and Jingjing Zhang. 2011. Maximizing stability of recommendation algorithms: A collective inference approach. In *21st Workshop on Information Technologies and Systems (WITS)*. Citeseer, 151–56.

[3] Gediminas Adomavicius and Jingjing Zhang. 2016. Classification, ranking, and top-K stability of recommendation algorithms. *INFORMS Journal on Computing* 28, 1 (2016), 129–147.

[4] Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*. Ieee, 39–57.

[5] Konstantina Christakopoulou and Arindam Banerjee. 2019. Adversarial attacks on an oblivious recommender. In *Proceedings of the 13th ACM Conference on Recommender Systems, RecSys 2019, Copenhagen, Denmark, September 16-20, 2019*, Toine Bogers, Alan Said, Peter Brusilovsky, and Domonkos Tikk (Eds.). ACM, 322–330. https://doi.org/10.1145/3298689.3347031

[6] Yashar Deldjoo, Tommaso Di Noia, and Felice Antonio Merra. 2021. A Survey on Adversarial Recommender Systems: From Attack/Defense Strategies to Generative Adversarial Networks. *ACM Comput. Surv.* 54, 2 (2021), 35:1–35:38. https://doi.org/10.1145/3439729

[7] Yali Du, Meng Fang, Jinfeng Yi, Chang Xu, Jun Cheng, and Dacheng Tao. 2018. Enhancing the robustness of neural collaborative filtering systems under malicious attacks. *IEEE Transactions on Multimedia* 21, 3 (2018), 555–565.

[8] Hui Fang, Danning Zhang, Yiheng Shu, and Guibing Guo. 2020. Deep learning for sequential recommendation: Algorithms, influential factors, and evaluations. *ACM Transactions on Information Systems (TOIS)* 39, 1 (2020), 1–42.

[9] Min Gao, Bin Ling, Quan Yuan, Qingyu Xiong, and Linda Yang. 2014. A robust collaborative filtering approach based on user relationships for recommendation systems. *Mathematical Problems in Engineering* 2014 (2014).

[10] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).

[11] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)* 5, 4 (2015), 1–19.

[12] Xiangnan He, Zhankui He, Xiaoyu Du, and Tat-Seng Chua. 2018. Adversarial personalized ranking for recommendation. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 355–364.

[13] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of the 27th ACM international conference on information and knowledge management*. 843–852.

[14] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).

[15] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*. IEEE, 197–206.

[16] Shigeki Karita, Nanxin Chen, Tomoki Hayashi, Takaaki Hori, Hirofumi Inaguma, Ziyan Jiang, Masao Someki, Nelson Enrique Yalta Soplin, Ryuichi Yamamoto, Xiaofei Wang, Shinji Watanabe, Takenori Yoshimura, and Wangyou Zhang. 2019. A Comparative Study on Transformer vs RNN in Speech Applications. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. 449–456. https://doi.org/10.1109/ASRU46091.2019.9003750

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

[18] Walid Krichene and Steffen Rendle. 2022. On sampled metrics for item recommendation. *Commun. ACM* 65, 7 (2022), 75–83.

[19] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 1419–1428.

[20] R Logesh, V Subramaniyaswamy, D Malathi, N Sivaramakrishnan, and Varadarajan Vijayakumar. 2020. Enhancing recommendation stability of collaborative filtering recommender system through bio-inspired clustering ensemble method. *Neural Computing and Applications* 32, 7 (2020), 2141–2164.

[21] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083* (2017).

[22] Jarana Manotumruksa and Emine Yilmaz. 2020. Sequential-based Adversarial Optimisation for Personalised Top-N Item Recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, Jimmy X. Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu (Eds.). ACM, 2045–2048. https://doi.org/10.1145/3397271.3401264

[23] Takeru Miyato, Andrew M Dai, and Ian Goodfellow. 2016. Adversarial training methods for semi-supervised text classification. *arXiv preprint arXiv:1605.07725* (2016).

[24] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. 2018. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence* 41, 8 (2018), 1979–1993.

[25] Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*. 188–197.

[26] John O'Donovan and Barry Smyth. 2005. Trust in recommender systems. In *Proceedings of the 10th international conference on Intelligent user interfaces*. 167–174.

[27] Sejoon Oh, Berk Ustun, Julian McAuley, and Srijan Kumar. 2022. Rank List Sensitivity of Recommender Systems to Interaction Perturbations. *31st ACM International Conference on Information and Knowledge Management (CIKM 2022)* (2022).

[28] Michael O'Mahony, Neil Hurley, Nicholas Kushmerick, and Guénolé Silvestre. 2004. Collaborative recommendation: A robustness analysis. *ACM Transactions on Internet Technology (TOIT)* 4, 4 (2004), 344–377.

[29] Zohreh Ovaisi, Shelby Heinecke, Jia Li, Yongfeng Zhang, Elena Zheleva, and Caiming Xiong. 2022. RGRecSys: A Toolkit for Robustness Evaluation of Recommender Systems. In *WSDM '22: The Fifteenth ACM International Conference on Web Search and Data Mining, Virtual Event / Tempe, AZ, USA, February 21 - 25, 2022*, K. Selcuk Candan, Huan Liu, Leman Akoglu, Xin Luna Dong, and Jiliang Tang (Eds.). ACM, 1597–1600. https://doi.org/10.1145/3488560.3502192

[30] Dae Hoon Park and Yi Chang. 2019. Adversarial sampling and training for semi-supervised information retrieval. In *The World Wide Web Conference*. 1443–1453.

[31] Aditi Raghunathan, Sang Michael Xie, Fanny Yang, John C. Duchi, and Percy Liang. 2019. Adversarial Training Can Hurt Generalization. *CoRR* abs/1906.06032 (2019). arXiv:1906.06032 http://arxiv.org/abs/1906.06032

[32] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*. 811–820.

[33] David Shriver, Sebastian Elbaum, Matthew B Dwyer, and David S Rosenblum. 2019. Evaluating recommender system stability with influence-guided fuzzing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 4934–4942.

[34] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*. 1441–1450.

[35] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199* (2013).

[36] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the eleventh ACM international conference on web search and data mining*. 565–573.

[37] Ahmet Murat Turk and Alper Bilge. 2019. Robustness analysis of multi-criteria collaborative filtering algorithms against shilling attacks. *Expert Systems with Applications* 115 (2019), 386–402.

[38] Fan Wang, Haibin Zhu, Gautam Srivastava, Shancang Li, Mohammad R Khosravi, and Lianyong Qi. 2021. Robust collaborative filtering recommendation with user-item-trust records. *IEEE Transactions on Computational Social Systems* (2021).

[39] Shoujin Wang, Liang Hu, Yan Wang, Longbing Cao, Quan Z Sheng, and Mehmet Orgun. 2019. Sequential recommender systems: challenges, progress and prospects. *arXiv preprint arXiv:2001.04830* (2019).

[40] Rey Reza Wiyatno, Anqi Xu, Ousmane Dia, and Archy de Berker. 2019. Adversarial Examples in Modern Machine Learning: A Review. *CoRR* abs/1911.05268 (2019). arXiv:1911.05268 http://arxiv.org/abs/1911.05268

[41] Chenwang Wu, Defu Lian, Yong Ge, Zhihao Zhu, Enhong Chen, and Senchao Yuan. 2021. Fight Fire with Fire: Towards Robust Recommender Systems via Adversarial Poisoning Training. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1074–1083.

[42] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. 2017. Recurrent recommender networks. In *Proceedings of the tenth ACM international conference on web search and data mining*. 495–503.

[43] Xu Xie, Fei Sun, Zhaoyang Liu, Shiwen Wu, Jinyang Gao, Jiandong Zhang, Bolin Ding, and Bin Cui. 2022. Contrastive learning for sequential recommendation. In *2022 IEEE 38th international conference on data engineering (ICDE)*. IEEE, 1259–1273.

[44] An Yan, Shuo Cheng, Wang-Cheng Kang, Mengting Wan, and Julian McAuley. 2019. CosRec: 2D convolutional neural networks for sequential recommendation. In *Proceedings of the 28th ACM international conference on information and*

*knowledge management*. 2173–2176.

[45] Feng Yuan, Lina Yao, and Boualem Benatallah. 2019. Adversarial collaborative neural network for robust recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1065–1068.

[46] Zhenrui Yue, Zhankui He, Huimin Zeng, and Julian McAuley. 2021. Black-Box Attacks on Sequential Recommenders via Data-Free Model Extraction. In *Fifteenth ACM Conference on Recommender Systems*. 44–54.

[47] Hengtong Zhang, Yaliang Li, Bolin Ding, and Jing Gao. 2020. Practical data poisoning attack against next-item recommendation. In *Proceedings of The Web Conference 2020*. 2458–2464.