“© 2019 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.”

Enhancing the Robustness of Neural Collaborative Filtering Systems Under Malicious Attacks

Yali Du, Meng Fang, Jinfeng Yi, Chang Xu, Jun Cheng, and Dacheng Tao, *Fellow, IEEE*

***Abstract*—Recommendation system has become ubiquitous in online shopping in recent decades due to its power in reducing excessive choices of customers and industries. Recent collaborative filtering methods based on the deep neural network are studied and introduce promising results due to their power in learning hidden representations for users and items. However, it has revealed its vulnerabilities under malicious user attack. With the knowledge of a collaborative filtering algorithm and its parameters, the performance of this recommendation system can be easily downgraded. Unfortunately, this problem is not addressed well, and the study on defending recommendation systems is insufficient.**

In this paper, we aim to improve the robustness of recommendation systems based on two concepts: *stage-wise hints training* and *randomness*. To protect a target model, we introduce noise layers in the training of a target model to increase its resistance to adversarial perturbations. To reduce noise layers' influence on model's performance, we introduce intermediate layers' outputs as hints from a teacher model to regularize the intermediate layers of a student target model. We consider white box attacks under which attackers have the knowledge of the target model. The generalizability and robustness properties of our method have been analytically inspected in experiments and discussions, and the computation cost is comparable to training a standard neural network-based collaborative filtering model. Through our investigation, the proposed defensive method can reduce the success rate of malicious user attacks and keep the prediction accuracy comparable to standard neural recommendation systems.

***Index Terms*—Recommendation Systems, Adversarial Learning, Collaborative Filtering, Malicious Attacks**

I. Introduction

In the recent decade, recommendation systems have become a pivotal tool for both industries and customers due to information explosion on the Internet. On the one hand, customers rely on recommendation systems to generate the list of items that are possibly favored by customers without skimming through the enormous choices [1], [2]. On the other hand, industries aim to build effective recommendation systems that can predict users' potential interests in unseeing items and thus make appropriate recommendations to generate revenue [3]. Successful recommendation systems are capable of correctly modeling users' favor over items based on their previous interactions, such as ratings, review comments [4]. The key to building this effective recommendation system is modeling the hidden factors of both users and items relying on human feedback [5], [6]. Matrix factorization has become one of the most successful approaches that map users and items into a shared hidden space, and then predict new ratings by the interaction of user and item's latent representations [6], [7]. Recently, deep learning algorithms are used to discover good representations of users and items due to their power in learning discriminative representations of different examples [8], [9], [10], [11], [12] and achieve good results.

By using these recommendation algorithms, such as factorization based approaches, the developers make an implicit assumption that they are secure. However, due to the open property of recommendation systems [13], [14], recent works show that factorization based approaches are vulnerable to malicious user attacks [15]. Malicious users are threatening artifacts to the recommendation systems that can be crafted if the adversary knows the architecture of the recommendation system or legitimate user data [15], [16], [17]. Attackers, on the one hand, try to craft malicious user data that degrade the system's performance, on the other hand, they will keep their behaviors close to normal users to avoid detection. In attacking a recommendation system, malicious users' goals are diversified. By carefully exploit from training data and the structure of recommendation systems, adversaries can mislead the system to produce the target value that is far from the ground truth to achieve their malicious goals. The general purpose for an adversary falls into two categories: to promote (or demote) the popularity of a specific set of items and decrease the overall accuracy of the target recommendation systems to downgrade users' trust on it [18]. One important rule for crafting malicious examples for collaborative filtering is to keep the adversarial data close to the legitimate data, making them hard to be detected by the human.

Different attacks such as random attacks and push attacks have been investigated [16], [17], [18]. While [16], [17] are both built upon the neighbor-based collaborative filtering system. [15] proposes a more aggressive data poisoning attack to decrease the general performance of collaborative filtering models which is based on full knowledge of the learning algorithm and training data. By solving a min-max objective

Y. Du is with Faculty of Engineering and Information Technology, University of Technology Sydney, 81 Broadway Street, Ultimo, NSW 2007, Australia and the UBTECH Sydney Artificial Intelligence Centre and the School of Information Technologies, the Faculty of Engineering and Information Technologies, the University of Sydney, 6 Cleveland St, Darlington, NSW 2008, Australia(e-mail: yali.du@student.uts.edu.au).

M. Fang is with the Tencent AI Laboratory, Shenzhen 518057, China (e-mail: moefang@gmail.com)

J. Yi is with JD AI Research, Beijing 100020, China (e-mail: yijinfeng@jd.com)

J. Cheng is with Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China (e-mail: jun.cheng@siat.ac.cn)

C. Xu and D. Tao are with the UBTECH Sydney Artificial Intelligence Centre and the School of Information Technologies, the Faculty of Engineering and Information Technologies, the University of Sydney, 6 Cleveland St, Darlington, NSW 2008, Australia (email: {c.xu, dacheng.tao}@sydney.edu.au).

function, the optimal set of malicious users can be reached to contaminate collaborative filtering system's performance. Despite these attack strategies on collaborative filtering systems, the defense approaches for collaborative filtering are limited [15]. A close line of established research in robust collaborative filtering lies in robust matrix completion [19], [20], [21] in which a few entries or columns are randomly perturbed. However, none of them handle maliciously contaminated features for collaborative filtering.

In this paper, we focus on designing an effective training paradigm for obtaining robust collaborative filtering models. We investigate the recent popular neural network based collaborative filtering algorithms and design defensive methods for trustworthy collaborative filtering. Inspired by knowledge distillation [22], [23], we consider to build our training paradigm by using knowledge from different layers of a teacher model to improve the training of student models. The standard collaborative filtering model is treated as a teacher. We use the similar structure to build a student collaborative filtering model. Extra noise layers are added before dense layers in a student model to improve its robustness to small perturbations. Note that the student model usually has the same or smaller size than the teacher model and the discrepancy between dimensionality of intermediate layers from teacher and student can be mitigated by a mapping function. By introducing intermediate representations from a teacher, we first make the student model to learn to predict the intermediate representations of the teacher, then train the whole student model. The noise layer enforces the student network to learn the neural layers that are robust to perturbations [**?**], and the hints training paradigm regularizes the student model to have similar performance to the teacher model.

We summarize the contributions of this paper in the following:

- We propose an effective training strategy to obtain robust neural network based collaborative filtering system. Moreover, our model is one of the first works addressing this problem.
- We analytically demonstrate that the proposed framework can reduce the system's sensitivity to adversarial gradients exploited by malicious users and retain accuracy in predictions.
- We experimentally verify that our proposed neural collaborative filtering algorithm performs to be less sensitive to adversarial perturbations generated by the state-of-the-art C&W attack than approaches that do not deploy our defensive training strategy and retain accuracy comparable to standard models.

The rest of this paper is structured as follows. In Section II, we discuss related works. We introduce preliminary knowledge of neural collaborative filtering approach and the attack strategy in Section III. The formalization of our method is presented in Section IV followed by analytical discussion in Section V. Empirical verifications are presented in Section VI and the paper is concluded in Section VII.

II. Related Works

In this section, we briefly review the collaborative filtering algorithms, the threatening factors to them and possible transferable robust training techniques from other applications.

Collaborative filtering contains two main group of approaches: nearest neighbor-based [24] and model-based approaches [25], [26]. For nearest neighbor-based algorithms, a user's (item's) preference is estimated from users (items) that have similar profiles with it. For model-based collaborative filtering, a parametric model is constructed and the estimated from the observed interactions between users and products, henceforth makes predictions on unseen entries. Model-based methods have been proved to be superior to neighbor-based methods, and among them, matrix factorization algorithms [27], [28], [29] are the most popular model-based collaborative filtering techniques. Factorization-based methods estimate the hidden user and item factors from their past interactions and make predictions by multiplication of estimated user and item factors. To boost the collaborative filtering performance, another established line of works make use of side information and construct inductive matrix completion model [30], [31].

An effective recommendation system heavily relies on accurately modeling the hidden representations for users and items. Due to the power of neural networks in representation learning, recently neural network-based collaborative filtering (NeuCF) becomes popular and claims promising performance on a set of benchmarks. Neural network-based collaborative filtering models take paired user and item features as inputs [9], [10]. Through feature embedding layers, user and item features are projected into their respective latent space ensuing by different neural network layers and activation functions. [10] utilizes bilinear decoder to map the embedded user and item factors to ratings, and [9] uses a linear classifier with sigmoid activation function. While autoencoder-based approaches [32] take either item or user feature as inputs, they can be treated as a special case of standard NeuCF architecture. [11], [33] share similar architectures to autoencoder-based models but utilizes a stochastic approach. Instead of ReLU or logistic units, [33] utilizes stochastic units with a particular distribution such as binary of Gaussian, while [11] proposes a more advanced model capable of handling high dimensional binary vectors.

While many works have been devoted to improving the performance of recommendation systems from different aspects, such as proposing new algorithms, employing various content information or side information, the security of collaborative filtering algorithms is not broadly studied. Below we first introduce attacking techniques ensuing by studies on defensive models.

In [16], [17], [18], researchers study attacks on neighbor-based collaborative filtering. [15] studies data poisoning attack on factorization-based recommendation systems, which attacks during the training phase of collaborative filtering. [34] shows that the accuracy and robustness reaches a balance. While they achieve superior performance in impairing existing algorithm's performance, the investigation of the provable defense strategy is limited. A trending topic that attracts huge attention is the vulnerabilities of deep neural networks (DNN) [35], [36], [37].

By adding some imperceptible perturbation to the image, the classification results DNN might vary a lot, owing to the finding in [35] that the smoothness assumption lying under many kernel methods does not hold. Different attacks claim overwhelming success in generating adversarial examples. To name a few, fast gradient sign method (FGSM) [36] generates adversarial perturbations in the direction of loss function gradients; Jacobian Saliency Map Attack (JSMA) [37] estimates the output's sensitivity to input feature dimensions and modifies the most impactive features until it varies the classification; and C&W attack solves an optimization problem to find malicious perturbed images. Through light modifications to these attacks, they can be applied to Neural collaborative filtering algorithm as well.

Compared with the adversarial attack, the defense is a much more challenging task, and fewer works have been devoted to this problem, especially in the area of recommendation systems. Distillation [38] is proposed as a defense to adversarial perturbations through training a student network guided by a teacher [39]. By training a network with soft labels, the distilled network will be prone to avoid over-fitting on training data, and thus be robust to maliciously perturbed inputs lying on the "blind spots" of non-linear networks. It can significantly reduce the effectiveness of adversarial examples on DNNs. However, Carlini and Wagner propose a powerful attack (C&W attack) that can still acquire high success rate on DNNs trained with defensive distillation [40]. Several variations includes [**?**], [41] follow a similar framework to C&W attack. The C&W attack has been recognized as the state-of-the-art white-box attack.

III. Preliminaries

We first revisit the matrix factorization (MF) in collaborative filtering algorithms and then neural network based collaborative filtering, which is the target model of attack and defense in this work. Then we describe adversarial attack details for the model in recommendation systems.

Notations: Throughout this paper, we consider histograms of dimension $n+1$. We use bold upper case letters $\mathbf{X}$ for matrices, and bold lower case letters $\mathbf{x}$ for vectors. Lower case letters stand for scalars in $\mathbb{R}$.

A. Neural Collaborative Filtering

Consider a recommendation system with N_i users and N_j items and let i, j indicate i-th user and j-th item respectively, we denote the observed rating value as r_{ij} such that $i \in [N_i], j \in [N_j]$ which indicates the interactions between users and items. Let $r_{ij} \in \{1, 2, ..., L\}$ and Ω denote the index set of m observations with $(i, j) \in \Omega$ and $\mathbf{R}$ denote the underlying rating matrix. One would be interested in finding hidden factors of users and items and predict the interactions between them in the following factorization form:

$$r_{ij} = \mathbf{u}_i \mathbf{v}_j^\top, \tag{1}$$

where root mean square error is often applied to measure the recovery distance between ground truth ratings and predicted ones. Based on the side information of customers and products,

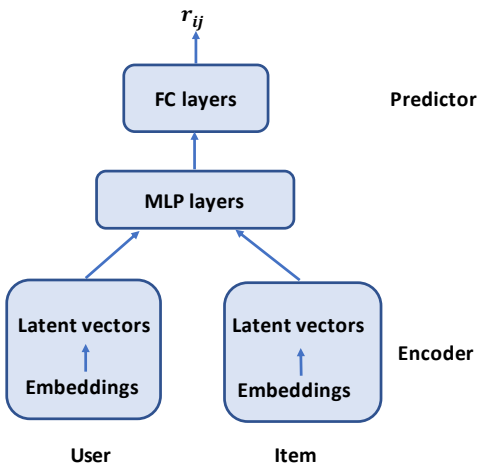

Fig. 1: Overview of a neural collaborative filtering architecture. It encodes user and item features, then uses multi-layer perceptrons to process their hidden representations, and generates rating predictions after fully-connected (FC) layers.

inductive matrix completion (IMC for short) assumes that ratings are generated by multiplying feature vectors of users and items to an unknown low-rank matrix. Based on this fact, rating predictions can be generated by $r_{ij} = \mathbf{x}_i \mathbf{M} \mathbf{y}_j^\top$, where $x_i \in \mathbb{R}^{d_x}$ and $y_j \in \mathbb{R}^{d_y}$ denote the features for i-th user and j-th item respectively, d_x and d_y are the dimension of user and item profile features. $\mathbf{M} \in \mathbb{R}^{d_x \times d_y}$ is the underlying matrix that aligns user and item side features.

Neural collaborative filtering is an extension based on classical collaborative filtering models. As shown in Figure 1, neural collaborative filtering takes user and item IDs p_i and q_j as input and encodes them to hidden representation as hidden factors. The embedding vectors for users and items are concatenated together and then feed to multi-layer perceptron layers. Denote the concatenated embeddings as π,

$$\pi(p_i, q_j) = \begin{bmatrix} \text{emb}(p_i) \\ \text{emb}(q_j) \end{bmatrix},$$

the neural collaborative filtering algorithm can be formulated as:

$$f(p_i, q_j) = \phi_{out}(\phi_X(...\phi_2(\phi_1(\pi(p_i, q_j)))...)), \tag{2}$$

where $\phi(\cdot)$ is dense layer with respective activation functions. Since neural collaborative filtering aims to predict implicit feedbacks between users and items, the output layer is a sigmoid activation function and $f(p_i, q_j)$ output the probability of if p_i is relevant to q_j. In this work, we make slight modification to the input by including side information together with user and item IDs. The concatenated embedding of users and items are now

$$\pi(p_i, q_j) = \begin{bmatrix} \text{emb}(p_i^{id}) \\ \text{emb}(p_i^{fea}) \\ \text{emb}(q_j^{id}) \\ \text{emb}(q_j^{fea}) \end{bmatrix}.$$

In the following, we will introduce attacks on the user and item's features to mislead the model to make wrong decisions.

B. Adversarial attacks

Previous works have shown that certain vulnerabilities of recommendation systems should be taken into consideration [15], [16], [17]. We consider the situation that adversary attacks a given neural collaborative filtering model to downgrade its overall performance. To answer how adversary craft fictitious user or item data to contaminate the model performance in the testing phase, we hence introduce the C&W attack [40] which has been recognized as the most powerful white-box attack.

For neural collaborative filtering system, we have two inputs from users and items. To jointly attack both user and item features, the adversary generally adds small perturbations $(\delta p_i, \delta q_j)$ into the input features such that $F(p_i + \delta p_i, q_j + \delta q_j) \neq r_{ij}$. C&W attack generates adversarial perturbations by solving an optimization problem. Due to the binary nature of implicit feedback prediction problem, we slightly changing the adversarial goal of C&W attack to make the NeuCF model make less confident predictions, which is formulated as:

$$\begin{aligned} &\min_{\delta p_i, \delta q_j} && \|\delta p_i\| + \|\delta q_j\| \\ &\text{s.t.} && f(p_i + \delta p_i, q_j + \delta q_j) \leq 0.5. \end{aligned} \tag{3}$$

This problem is equivalent to the following form:

$$\min_{\delta p_i, \delta q_j} \|\delta p_i\| + \|\delta q_j\| + c \cdot f(p_i + \delta p_i, q_j + \delta q_j) \tag{4}$$

where c is a suitable constant. The norm in the objective is specified depending on the perturbation shapes that adversary chooses. The perturbations can be small variations on all dimensions or sufficient alternations on a selected group of feature dimensions. Therefore, L_1, L_2 and L_∞ norm can be employed on the objective in Equation (3). Since it is easier to perturb the user's features than the item's in practice, it will not be necessary to perturb p_i and q_j at the same time. Notice that C&W attack is a white-box attack algorithm in which both the architecture and weights in F are known. An adversary can thus solve Equation (4) by optimization methods.

IV. Stage-wise hints training for robust neural collaborative filtering

Our approach contains two main components: stage-wise hints training and randomness. We design a teacher-student architecture based on knowledge transfer. The student model is similar to the structure of the teacher and but noise layers are deployed before each dense layer. First, we train basic collaborative filtering as a teacher. Second, we train a student model stage-wisely with different hints, which are intermediate representations from the teacher. Last, we train the whole student network with knowledge distillation.

A. Knowledge transfer from a teacher

Inspired by distillation network [22], we start our method with a basic model named as a teacher. We follow the same structure as used in [9] and include side information to construct a feature-based neural collaborative filtering network that takes user and item information as input and output implicit feedback of whether (user, item) pairs have interaction before.

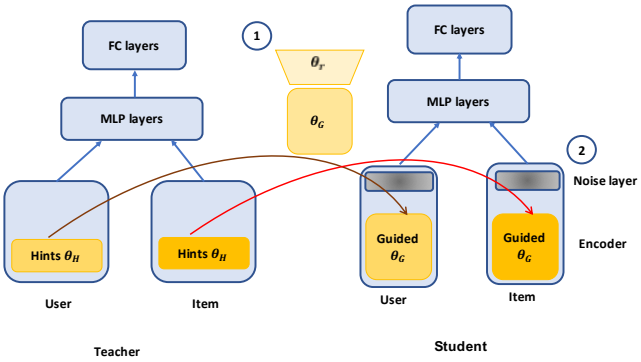

Fig. 2: The graph illustration of student training procedure: (i) Step 1 is the hints training process which aligns the output between intermediate layers of teachers and students; θ_r denotes the regression parameters that align the output dimension of teacher and students; (ii) Step 2 is the knowledge distillation training process with noise layers added before MLP layers.

At first, we choose a hidden layer of the encoder of the teacher and obtain the output of this layer and use this output as the knowledge for transferring. The knowledge is also called hints because it allows a student network to mimic it [23]. In this way, the student network is guided by the knowledge from the teacher. The knowledge is constructed by using the hints module, defined as

$$h_{k_i} = \text{emb}_H(p_i, q_j, \theta_H), \tag{5}$$

where θ_H denotes the parameters for hints module which is a subnet of teacher network up to the respective hidden layer, and emb_H indicates the corresponding deep nested functions. h_{k_i} denotes hints knowledge and k_i is the index of intermediate layer.

We also have a student model, which has a similar architecture with the teacher but it can be a thinner and deeper network than the teacher. We transfer the knowledge h_{k_i} from the teacher to the student. We use this knowledge as the output of a hidden layer of the encoder of the student to control how to train the student. Thus the training of the student network is guided by this knowledge. When training the student, there is a guided module in the student. This part is controlled by h_{k_i}, that is

$$\mathcal{L}(\theta_G, \theta_r) = \frac{1}{2}||h_{k_i} - \sigma(\theta_r \cdot \text{emb}_G(p_i, q_j, \theta_G) + b)||. \tag{6}$$

We add a regressor parameterized by θ_r on top of the guided layer of the student in case that the sizes of the student and teacher are different.

Figure 2 shows the hints transfer from embedding layers of a teacher to a student before feeding to Multi-layer perceptron (MLP). The teacher has a larger dimension for user and item's embeddings. Thus θ_r is introduced to map the student model's embedding to the space of the teacher model's embedding.

Finally, based on pre-trained parameters in guided layers, we train the parameters of the whole student network with

knowledge distillation. The distilled model is trained by the following objective function:

$$\min_{\theta_F} -\frac{1}{|\Omega|}\sum_{(i,j)\in\Omega}\sum_{l\in\{1,...,L\}} F_l \log F_l^d(p_i,q_j), \quad (7)$$

where F is the output of softmax layer of teacher model, F^d indicates that of student model and θ_F is the whole set of parameters for F. $\{1,...,L\}$ denotes the label set. As we predict implicit feedbacks, 1 indicates p_i is relevant to q_j and 0 otherwise [9]. We have binary output and the objective function for distillation is structured as following:

$$-\frac{1}{|\Omega|}\sum_{(i,j)\in\Omega}(1-F_1)\log(1-F_1^d(p_i,q_j))F_1\log F_1^d(p_i,q_j), \quad (8)$$

If the amplitude of adversarial gradients is high, executing cost of adversary will be low, since altering the input, the variations of outputs will vary a lot. In this way, a good way to protect the model from being attacked is to reduce the variations around inputs.

B. Stage-wise hints training of the student model

Rather than learn from only the softmax layer or an intermediate layer of the teacher, we train the student model stage-wisely based on hints from different layers of the teacher. We allow different hints obtained from the same teacher, which is different from the usage of distillation in previous works. Based on Equation (5), we choose different size of modules and then obtain different hints $h_1,\cdots,h_m$ corresponding to these modules respectively. For stage-wise training, the guided module in each stage in the student model is different and follows the rule from shallow to deep.

We summarize the stage-wise training regime in Algorithm 1. By setting a standard collaborative filtering model as the teacher, we choose m hints modules from teacher model and m guided modules from the student model. Then we update the student network by m stages. We train the student model from shallow to deep, which implies that $g_1 < g_2 < ... < g_m$ and $h_1 < h_2 < ... < h_m$. After stage-wise hints training, the student model is acquainted with knowledge from the teacher and can predict the intermediate representations of the teacher model.

C. Randomness by injecting noises

As mentioned in Section IV-A, to protect the model from being attacked one need to reduce the variations around inputs. This is not easy to be controlled during white-box attacks. On the contrary, we train the collaborative filtering model that is robust to variations of inputs. We consider improving the robustness of neural networks by adding randomness into the network structure. In particular, we introduce a new "noise layer" that fuses input vector with a randomly generated noise, defined as

$$\theta_S^i \leftarrow \theta_S^i + \epsilon, \quad \epsilon \sim \mathcal{N}(0,\delta^2), \quad (9)$$

where θ_S^i is i-th layer student model. If $\epsilon = 0$, no noise is introduced. If $\epsilon > 0$, through minimizing Equation (6), we are looking for a student model that spans the same space of activations of teacher model but is robust to perturbations as well.

Algorithm 1 A sketch of stage-wise hints training based on a pre-trained teacher model

Input: teacher network θ_T, student network θ_S, $g_1,...,g_m$, $h_1,...,h_m$
Output: student model acquainted of transferred knowledge θ_S^*

for m hints modules **do**
  $\theta_H \leftarrow \{\theta_T^1,...,\theta_T^{h_i}\}$
  $\theta_G \Leftarrow \{\theta_S^1,...,\theta_S^{g_i}\}$
  initialize θ_r
  Choose a intermediate layer from the teacher T and generate a hint h_m
  Train the guided layers of the student using h_m
  $\theta_G^* = \arg\min_{\theta_G} \mathcal{L}(\theta_G,\theta_r)$
  $\{\theta_S^1,...,\theta_S^{g_i}\} = \{\theta_G^{*1},...,\theta_G^{*g}\}$
end for

The noise introduced in training time is crucial for increasing the target model's robustness to perturbations. Through hints training strategy, the guided layer will be robust to the perturbations as well as produce similar results to the hints layer. The final step is training the student model by knowledge distillation strategy. Figure 3 shows the framework of the proposed method. The teacher model generates several useful hints and distills the knowledge to student model by different stages.

D. Relation to other works

The stage-wise hints training with knowledge distillation can be seen as a particular form of curriculum learning [42]. By choosing a proper sequence of tasks to learning, curriculum learning is proved to accelerate the convergence and improve generalization. In this work, we use a teacher model as a guide and train the student model from simple to complex ones, which would considerably ease training [43]. [23] adopts a single stage hints training framework which can be seen as a special case of our method. While [23] uses hints to guide the training of a thinner and deeper network, we use hints training together with randomness to increase model's robustness to perturbations.

Except for the training of the teacher model, our method does not introduce computation overhead on training student model. Though we train the student model in multi-stages, the overall epochs of training are not increased much. If a well-performed neural collaborative filtering model has already existed, the cost for training our robust neural collaborative filtering model can be further reduced.

V. Analysis of stage-wise hints training strategy

In this section, we first establish the *robustness* definition for neural collaborative filtering model and illustrate the vulnerability of NeuCF. Then we analytically investigate the impact of our multiple hints training strategy to the neural collaborative

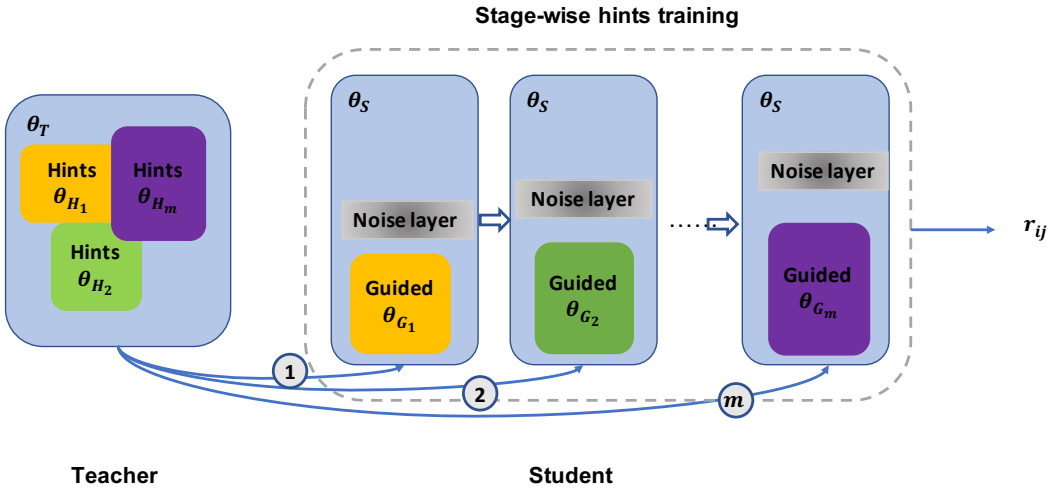

Fig. 3: The stage-wise hints training framework. Steps 1 to m are the hints training routine that is illustrated in Figure 2 but need to be repeated m times for m different hints modules from the teacher.

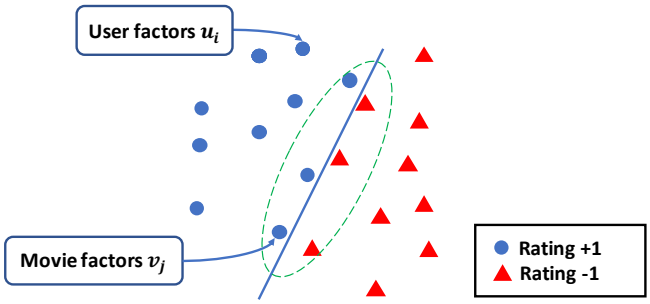

Fig. 4: The graph illustration of vulnerabilities of the collaborative filtering system

filtering system from two perspectives: (i) model's sensitivity to adversarial perturbations, (ii) generalizability of neural collaborative filtering model.

A. *Robustness of neural collaborative filtering system*

The *robustness* of a neural collaborative algorithm to adversarial perturbations is defined as its capacity against perturbations. A robust neural collaborative filtering algorithm should (i) show good prediction accuracy in the training set and testing set as well as (ii) model a smooth neural collaborative filtering machine that can make stable predictions around the neighborhood of a legitimate user. The definition of *robustness* introduced in [38] can be slightly modified to the collaborative filtering here, which is:

$$\rho_{adv}(F) = E_{\mu}[\Delta_{adv}(p_i, q_j, F)] \tag{10}$$

where inputs (i, j) are drawn from an unknown distribution μ that neural collaborative filtering algorithm attempt to model with function F. $\Delta_{adv}(i, j, F)$ is defined as the minimal perturbations needed to wrongly predict sample (i, j):

$$\Delta_{adv}(i, j, F) = \arg\min_{\delta p_i, \delta q_j} \{\|\delta p_i\| + \|\delta q_j\| : F(p_i + \delta p_i, q_j + \delta q_j) \neq F(p_i, q_j)\} \tag{11}$$

where $\|\cdot\|$ is the norm that need to be specified according to systems' requirements. A robust neural collaborative filtering system requires higher average minimum perturbations to wrongly predict an example from distribution μ.

To illustrate the vulnerability of neural collaborative filtering algorithm, we consider using a bilinear decoder [10] to predict a user's preference on an item. Assume user and item features are embedded into $\mathbf{u}_i = \text{emb}_{user}(p_i), \mathbf{v}_j = \text{emb}_{item}(q_j)$, the prediction function is as follows:

$$p(r_{ij} = l) = \frac{e^{\mathbf{u}_i Q_l \mathbf{v}_j^\top}}{\sum_{l=1}^{L} e^{\mathbf{u}_i Q_l \mathbf{v}_j^\top}}. \tag{12}$$

If user feature factor U is fixed, the collaborative filtering problem can be taken as a linear classification problem for each item v_j. In Figure 4, we show a possible solution for given positive and negative ratings in red and blue color. If we treat item factor v_j as the classifier indicated by the straight line in Figure 4 and the blue dots and red triangles as user feature vectors, then predicting their interactions labeled as "+1" or "-1" can be treated as a binary classification problem. In Figure 4, although this is a correct solution which classifies both positive and negative ratings correctly, we can see that it is vulnerable to attacks since six examples in green eclipse are very close to the decision boundary. By slightly perturb the input user feature p_i or item features q_j, the six points in the green eclipse will be wrongly predicted.

Note that we consider a linear case to make the intuition simple to be understood. In other cases, the hyperplane would be more complexed. Figure 4 indicates that to let positively (negatively) labeled user have a larger neighborhood with same

labels will lead to a more robust neural collaborative filtering system. During training the NeuCF model, our goal is to converge to a function F^* that is against adversarial perturbations and generalize well. According to the universality theorem proposed in [44] for neural networks, the existence of such F^* is guaranteed. The hints training technique is thus deployed in the network training and helping it converge to the optimal F^* without getting stuck into local optima.

B. Impact of stage-wise hints training on model sensitivity to inputs

Through our hints training strategy, the model's sensitivity to malicious perturbations is reduced. We then derive the distilled model's sensitivity to input features. Denote $g(p_i, q_j) = \sum_{h=1}^{L} e^{z_h(p_i,q_j)/T}$, we then have that

$$\begin{aligned}
&\frac{\partial F_l(p_i,q_j)}{\partial p_{ik}} = \frac{\partial}{\partial p_{ik}}\left(\frac{e^{z_l/T}}{\sum_{h=1}^{L} e^{z_h/T}}\right) \\
=&\frac{1}{g^2(p_i,q_j)}\left(\frac{\partial e^{z_l/T}}{\partial p_{ik}} g(p_i,q_j) - e^{z_l/T}\frac{\partial g(p_i,q_j)}{\partial p_{ik}}\right) \\
=&\frac{1}{g^2(p_i,q_j)}\frac{e^{z_l/T}}{T}\left(\sum_{h=1}^{L}\frac{\partial z_l}{\partial p_{ik}} e^{z_h/T} - \sum_{h=1}^{L}\frac{\partial z_h}{\partial p_{ik}} e^{z_h/T}\right) \\
=&\frac{1}{T}\frac{e^{z_l/T}}{g^2(p_i,q_j)}\left(\sum_{h=1}^{L}\left(\frac{\partial z_l}{\partial p_{ik}} - \frac{\partial z_h}{\partial p_{ik}}\right) e^{z_h/T}\right). \qquad (13)
\end{aligned}$$

The last equation shows that Jacobian decreases with increasing temperature T because Jacobian is inversely proportional to T, and outputs of last hidden layer are divided by T before exponentiated.

This analysis shows that by using high temperature T in the network, the amplitude of adversarial gradients change is reduced with the variation of inputs. It is important to notice that the high temperature T only changes the amplitude of logit but not the relative orders of these logits, which does not influence the prediction accuracy. At test time, temperature T is set to $T = 1$, which will lead to more discrete rating predictions.

If we have a look into the hints training formulated by Equation (6), hints from teacher network plays as a regularization term that constrains the guided module make the correct prediction and being robust to random perturbations simultaneously.

The proposed training strategy has several superiorities. Firstly, the distillation technique only smooths the amplitude of logits but not change the relative order of different logits, thus does not reduce the model's accuracy. Secondly, the distillation technique has a low impact on the model's architecture while proposing a new architecture requires much effort in analyzing its effectiveness. Furthermore, the slight modification to the training procedure does not reduce the model's speed in the testing phase. By introducing the distilled model, a low computation cost of training a teacher model is added to the training phase, but it is a fixed cost and is acceptable.

C. Impact of stage-wise hints training on model's generalizability

When training a student network with noise added to its guiding layer, we are optimizing the following problem:

$$\theta^* = \arg\min_{\theta} \mathbb{E}_{\epsilon\sim\mathcal{N}(0,\sigma^2)}\mathcal{L}(F_{\theta,\epsilon}(p_i,q_j), r_{ij}). \qquad (14)$$

Based on the obtained parameters θ, we make predictions for ratings using the following form:

$$\hat{r}_{ij} = \arg\max \mathbb{F}_{\theta}(p_i, q_j). \qquad (15)$$

To explain why minimizing Equation (14) is able to yield similar prediction to original network, according to [**?**], we show that minimizing Equation (14) is equal to minimize the upper bound of the inference loss. Specifically, we have:

$$\begin{aligned}
&\mathbb{E}_{\epsilon\sim\mathcal{N}(0,\sigma^2)}\mathcal{L}(F_{\theta,\epsilon}(p_i,q_j), r_{ij}) \\
=&-\mathbb{E}_{\epsilon\sim\mathcal{N}(0,\sigma^2)}\log F_{\theta,\epsilon}(p_i,q_j)\cdot[r_{ij}] \\
\geq&-\log\mathbb{E}_{\epsilon\sim\mathcal{N}(0,\sigma^2)}F_{\theta,\epsilon}(p_i,q_j)\cdot[r_{ij}] \\
\geq&-\log\mathbb{E}_{\epsilon\sim\mathcal{N}(0,\sigma^2)}F_{\theta,\epsilon}(p_i,q_j)\cdot[\hat{r}_{ij}], \qquad (16)
\end{aligned}$$

where the first inequation holds because of Jensen's inequality and second inequation comes from Equation (15). This validates that minimizing Equation (14) is equal to minimizing the upper bound of $-\log F_{\theta,\epsilon}(p_i,q_j)[\hat{r}_{ij}]$ which is the inference loss.

VI. Experiments

In the following, we empirically evaluate the effectiveness of the proposed robust training strategy against adversarial attacks.

A. Dataset information

Since reviewing items is an onerous process in real recommendation systems, items that have more ratings from users are implicitly more popular than these that own fewer ratings. To simplify the discussion of our robust training strategy, we focus on this binary implicit feedback prediction problem. In this case, the untargeted and targeted attacks are equal. The attack aims to downgrade the overall performance of the collaborative filtering system. We experiment with two publicly available dataset MovieLens-1M and MovieLens-100K [**?**]. MovieLens-1M[1] includes 6,040 users, 3,952 movies and 1,000,209 anonymous ratings among them. We follow the setting in [9] which use MovieLens to investigate the implicit signal from explicit rating feedback. The one million ratings are transformed into implicit feedback indicating if the user has rated the corresponding item. This dataset include information for both users and items, such as "gender", "age", "occupation" for users and "genre", "release date" for movies.

MovieLens-100K[2] includes 100,000 ratings by 943 users on 1682 items. Each user has rated at least 20 movies. This dataset contains the same kind of side information as MovieLens-1M and we testify the collaborative filtering model's performance on predicting implicit feedbacks between users and items.

[1] http://grouplens.org/datasets/movielens/1m/
[2] https://grouplens.org/datasets/movielens/100k/

TABLE I: Overview of architectures of the teacher and student model. The second last dense layer determines the latent factor for the neural collaborative filtering model which is 64 and 32 for teacher and student model in this table.

Layer Type	Teacher	Student
Embedding	64 units	32 units
Embedding	64 units	32 units
Dense	64 units	32 units
Dense	64 units	32 units
Concatenate layer	256 units	128 units
ReLU Dense Layer	128 units	64 units
ReLU Dense Layer	64 units	32 units
ReLU Dense Layer	1 unit	1 unit
Sigmoid Activation	1 unit	1 unit

TABLE II: Overview of hyperparameters for training the model. For two stage hints training of student model, we train the model by 10 epochs in each stage and 10 epocs in final knowledge distillation training.

Parameter	Teacher	Student
Learning Rate	0.001	0.001
Optimizer	Adam	Adam
Batch Size	128	128
Total Epochs	20	30
Temperature	10	10

B. Overview of the experimental setup

To evaluate the proposed hints training strategy, we will compare its influence on model's generalizability to testing examples and its robustness to malicious attacks. For the neural collaborative filtering model, we develop a feature-based neural collaborative filtering model (FNCF) based on multi-layer percepetron model (MLP[3]). We follow a similar structure of that used in [9] and deploy a model with three dense layers. For latent factor as 64 and 32 of teacher and student model, the details of the architecture is shown in Table I. The teacher and student model share similar structure but the teacher model has larger latent factor which is the input dimensionality of the last dense layer. Both of them takes four inputs, which are user IDs, item IDs, user side information and item side information. The temperature for distillation is set to 10 through all the experiments. The set of hyper-parameters are presented in Table II.

1) Baseline setup: For the baseline of robust training strategy, we adopt *distillation as a defense* studied in [38]. The robust training strategy are listed as follows:

- FNCF-Distill trains a neural collaborative filtering model with knowledge distillation;
- FNCF-Single is a special case of our stage-wise hints training under which only the middle layer's representations of teacher model serve as hints to train the student model which is 128 units of a ReLU dense layer in Table I. Distillation training is deployed after single hint training.
- FNCF-Multi trains a model with two hints module and knowledge distillation. The hints modules produce output at first and second ReLU dense layer which are at length of 128 and 64 respectively as shown in Table I.

[3]https://github.com/hexiangnan/neural_collaborative_filtering

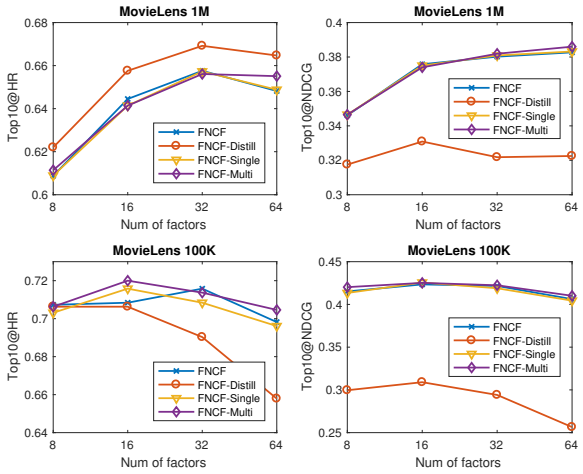

Fig. 5: Comparisons of FNCF with FNCF-Single, FNCF-Multi and FNCF-Distill at temperature $T = 10$ and noise level $\sigma = 0.05$.

To train the specified model, we use Keras [45] with Tensorflow [46] as the backend, which simplifies the implementation of architectures with multi-input and multi-output. The adversarial sample crafting is implemented by Tensorflow [46] based on pre-trained Keras models.

2) Evaluation criteria: To evaluate the performance of predictions on implicit feedbacks, we adopt the popular *leave-one-out* evaluation [9], [47] which construct the validation and test set by holding out one record for each user respectively. To judge the ranking list generated by collaborative filtering algorithms, we utilize Hit Ratio (HR) which is a recall-based metric and Normalized Discounted Cumulative Gain (NDCG) [48] with the rank list length set to be $K = 10$. HR is formulated by:

$$HR@K = \frac{Number\ of\ hits@K}{N}, \tag{17}$$

where N is the size of test set. Since HR@K is a recall-based metric, it does not reflect the model's capability of getting the top ranks correct. NDCG assigns higher score to results on top ranks, which is calculated by:

$$NDCG@K = Z_K \sum_{i=1}^{K} \frac{2^{r_i} - 1}{\log_2(i+1)}. \tag{18}$$

3) Adversarial example crafting: The adversarial goal of adversary is to alter the input (p_i, q_j) of a neural collaborative filtering model F to make the model F's prediction deviate from its groundtruth value. We use the C&W attack[4], with the L_2 norm regularization on perturbation variable δ and make slightly changes to adapt to our model with paired input. Note that as instructed by C&W attack, we take the output of logits layer to implement attacks, which is possible to circumvent the distillation as the defense.

[4]https://github.com/carlini/nn_robust_attacks

TABLE III: Different models' performance under different noise levels at temperature $T = 10$.

	MovieLens 1M									
	Top10@HR					Top10@NDCG				
Noise level	0	0.01	0.05	0.1	0.2	0	0.01	0.05	0.1	0.2
FNCF	0.6576	0.6576	0.6576	0.6576	0.6576	0.3802	0.3802	0.3802	0.3802	0.3802
FNCF-Single	0.6579	0.6575	0.6575	0.6584	0.6591	0.3797	0.3814	0.3821	0.3818	0.3818
FNCF-Multi	0.6548	0.6538	0.6571	0.6573	0.6591	0.3805	0.3799	0.3824	0.3823	0.3835
	MovieLens 100K									
	Top10@HR					Top10@NDCG				
Noise level	0	0.01	0.05	0.1	0.2	0	0.01	0.05	0.1	0.2
FNCF	0.7158	0.7158	0.7158	0.7158	0.7158	0.4216	0.4216	0.4216	0.4216	0.4216
FNCF-Single	0.7116	0.7126	0.7137	0.7147	0.7126	0.4201	0.4189	0.4216	0.4203	0.4179
FNCF-Multi	0.7126	0.7137	0.7094	0.7147	0.7094	0.4200	0.4214	0.4188	0.4198	0.4195

TABLE IV: Model's robustness against adversarial perturbations under different noise levels at temperature $T = 10$.

	MovieLens 1M									
	Success rate					L_2 distortion				
Noise level	0	0.01	0.05	0.1	0.2	0	0.01	0.05	0.1	0.2
FNCF	0.7320	0.7320	0.7320	0.7320	0.7320	0.8415	0.8415	0.8415	0.8415	0.8415
FNCF-Distill	0.7210	0.7210	0.7210	0.7210	0.7210	0.9790	0.9790	0.9790	0.9790	0.9790
FNCF-Single	0.7870	0.7250	0.6730	0.6320	0.6000	0.8446	0.8869	0.8210	0.7715	0.7808
FNCF-Multi	0.7810	0.7930	0.6420	0.6160	0.6120	0.8676	0.9155	0.8502	0.7318	0.7341
	MovieLens 100K									
	Success rate					L_2 distortion				
Noise level	0	0.01	0.05	0.1	0.2	0	0.01	0.05	0.1	0.2
FNCF	0.7783	0.7783	0.7783	0.7783	0.7783	1.0716	1.0716	1.0716	1.0716	1.0716
FNCF-Distill	0.7653	0.7653	0.7653	0.7653	0.7653	1.1173	1.1173	1.1173	1.1173	1.1173
FNCF-Single	0.7300	0.7050	0.6183	0.5800	0.5750	1.1183	1.0831	0.9994	0.9746	1.0004
FNCF-Multi	0.7200	0.7100	0.7000	0.6583	0.6533	1.0821	1.0645	1.0622	1.0313	1.0379

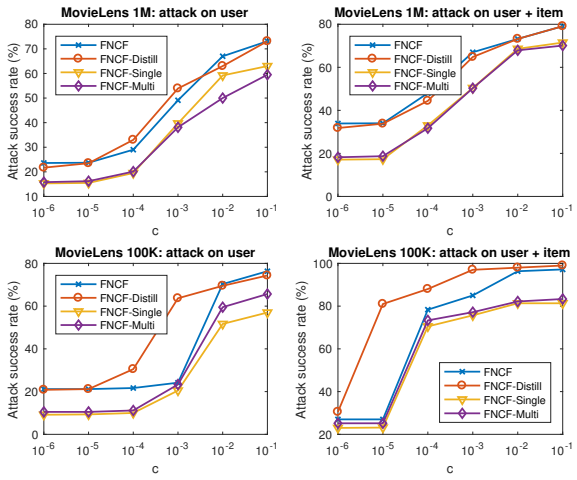

Fig. 6: Attack success rate on FNCF, FNCF-Single, FNCF-Multi and FNCF-Distill at temperature $T = 10$ and noise level $\sigma = 0.05$.

C. Influence on model's generalizability

In this set of experiments, we compare FNCF's performance with three training strategies under a different number of hidden factors. As shown in Figure 5, we observe that FNCF-Multi and FNCF-Single reach comparable performance to FNCF on both datasets. This verifies that the noise layer in FNCF-Multi and FNCF-Single will not introduce significant decreases in model's generalization. FNCF-Distill achieves disparate performance than the other three methods. Although FNCF-Distill has significant improvement on HR@K on MovieLens-1M, it performs poorly in terms of NDCG. This reflect that FNCF-Distill cannot make confident predictions on users and items' relations.

Furthermore, we quantify the noise's influence on the model's generalizability. In Table III, we vary the noise levels and observe that there are no significant variations between FNCF and FNCF-single, FNCF-Multi. Introducing noises into FNCF-Single renders a few variations to the prediction performance, which is acceptable. One can claim that the variations introduced by hints training process are moderate and can be neglected.

D. Impact on adversarial perturbations

In this set of experiments, we evaluate our training strategy's influence on malicious attacks by adversaries. For the examples to be attacked, we first select the examples that hit the top-10 list in predictions which are about 3900 for MovieLens-1M and 675 for MovieLens-100K. Then we randomly select 1000 examples from MovieLens-1M and all 675 examples from MovieLens-100K as the examples to be changed. We implement C&W attack on "user" and "user+item" which is discussed in Section III-B. For FNCF-Single and FNCF-Multi, the noise level is set to $\sigma = 0.05$. We report the success rate of C&W attack on different models under different const c specified in Equation (4). From Figure 6, we observe that FNCF-Distill does not show effectiveness on resisting adversarial attacks. FNCF-Single and FNCF-Multi reports lower attack success rates compared to baselines. This verifies the effectiveness of the proposed hints training algorithm.

E. Model's robustness under different noise levels

In this battery of experiments, we evaluate the model's robustness against malicious perturbations. The robustness measurement is defined in Equation (10). To exhaustively search over the entire input space is impossible. We average the distortions on examples that are successfully attacked. As shown in Table IV, with the increase of noise levels, the success rate of attack on both FNCF-Single and FNCF-Multi decreases, which validate the effectiveness of our robust training algorithm. The resulting L_2 distortion is also reported in Table IV. Notice that when no noise is introduced in $\sigma = 0$, FNCF-Single is more robust than FNCF-Distill. We credit this improvement to the hints training step that regularizes the student network.

VII. Conclusion

We have proposed a robust training strategy for neural collaborative filtering systems to improve its robustness to adversarial perturbations. We first use multiple hints to guide the training of multiple students. We also add noise layers in the student models to obtain robust units that are not sensitive to perturbed inputs. By ensembling multiple students with noise layers, we obtain the benefits that recommendation performance is not degraded, and the model's robustness is improved. Our experiments on benchmark datasets emphasize that defensive method with mixture models and noise training can still work even under current strong attackers.

References

[1] L. Sun, X. Wang, Z. Wang, H. Zhao, and W. Zhu, "Social-aware video recommendation for online social groups." *IEEE Transactions on Multimedia*, vol. 19, no. 3, pp. 609–618, 2017.

[2] Z. Xu, L. Chen, Y. Dai, and G. Chen, "A dynamic topic model and matrix factorization-based travel recommendation method exploiting ubiquitous data," *IEEE Transactions on Multimedia*, vol. 19, no. 8, pp. 1933–1945, 2017.

[3] Y. Zhou, J. Wu, T. H. Chan, S. wai Ho, D.-M. Chiu, and D. Wu, "Interpreting video recommendation mechanisms by mining view count traces," *IEEE Transactions on Multimedia*, 2017.

[4] J. Mcauley, "Hidden factors and hidden topics : Understanding rating dimensions with review text," in *RecSys*, 2013.

[5] J. Bennett and S. Lanning, "The netflix prize," in *Proceedings of KDD Cup and Workshop*, vol. 2007. New York, NY, USA, 2007, pp. 1–4.

[6] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, no. 8, pp. 30–37, 2009.

[7] Y. Du, C. Xu, and D. Tao, "Privileged matrix factorization for collaborative filtering," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. AAAI Press, 2017, pp. 1610–1616.

[8] H. Wang, N. Wang, and D.-Y. Yeung, "Collaborative deep learning for recommender systems," in *ACM International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 1235–1244.

[9] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2017, pp. 173–182.

[10] R. v. d. Berg, T. N. Kipf, and M. Welling, "Graph convolutional matrix completion," *KDD Deep Learning Day*, 2018.

[11] Y. Zheng, B. Tang, W. Ding, and H. Zhou, "A neural autoregressive approach to collaborative filtering," in *International Conference on Machine Learning*, 2016, pp. 764–773.

[12] Y. Wang, X. Lin, L. Wu, and W. Zhang, "Effective multi-query expansions: Collaborative deep networks for robust landmark retrieval," *IEEE Transactions on Image Processing*, vol. 26, pp. 1393–1404, 2017.

[13] G. Liu, Q. Chen, Q. Yang, B. Zhu, H. Wang, and W. Wang, "Opinionwalk: An efficient solution to massive trust assessment in online social networks," in *INFOCOM IEEE Conference on Computer Communications*. IEEE, 2017, pp. 1–9.

[14] Q. Yang and H. Wang, "Toward trustworthy vehicular social networks," *IEEE Communications Magazine*, vol. 53, no. 8, pp. 42–47, 2015.

[15] B. Li, Y. Wang, A. Singh, and Y. Vorobeychik, "Data poisoning attacks on factorization-based collaborative filtering," in *Advances in Neural Information Processing Systems*, 2016, pp. 1885–1893.

[16] M. P. O'Mahony, N. J. Hurley, and G. C. Silvestre, "Promoting recommendations: An attack on collaborative filtering," in *International Conference on Database and Expert Systems Applications*. Springer, 2002, pp. 494–503.

[17] B. Mobasher, R. Burke, R. Bhaumik, and C. Williams, "Effective attack models for shilling item-based collaborative filtering systems," in *Proceedings of the 2005 WebKDD Workshop*, vol. 2005, 2005, pp. 13–21.

[18] S. K. Lam and J. Riedl, "Shilling recommender systems for fun and profit," in *Proceedings of the 13th International Conference on World Wide Web*. ACM, 2004, pp. 393–402.

[19] Y. Chen, H. Xu, C. Caramanis, and S. Sanghavi, "Robust matrix completion and corrupted columns," in *Proceedings of the 28th International Conference on Machine Learning*, 2011, pp. 873–880.

[20] Y. Chen, A. Jalali, S. Sanghavi, and C. Caramanis, "Low-rank matrix recovery from errors and erasures," *IEEE Transactions on Information Theory*, vol. 59, no. 7, pp. 4324–4337, 2013.

[21] F. Nie, H. Wang, X. Cai, H. Huang, and C. Ding, "Robust matrix completion via joint schatten p-norm and lp-norm minimization," in *IEEE 12th International Conference on Data Mining (ICDM)*. IEEE, 2012, pp. 566–574.

[22] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *NIPS Deep Learning and Representation Learning Workshop*, 2015. [Online]. Available: http://arxiv.org/abs/1503.02531

[23] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "Fitnets: Hints for thin deep nets," *International Conference on Learning Representations*, 2015.

[24] G. Linden, B. Smith, and J. York, "Amazon. com recommendations: Item-to-item collaborative filtering," *IEEE Internet computing*, vol. 7, no. 1, pp. 76–80, 2003.

[25] J. D. Rennie and N. Srebro, "Fast maximum margin matrix factorization for collaborative prediction," in *Proceedings of 22nd International Conference on Machine Learning*. ACM, 2005, pp. 713–719.

[26] Y. Du, C. Xu, and D. Tao, "Collaborative rating allocation," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, 2017, pp. 1617–1623.

[27] K. Yu, S. Zhu, J. Lafferty, and Y. Gong, "Fast nonparametric matrix factorization for large-scale collaborative filtering," in *Proceedings of the 32nd international ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 2009, pp. 211–218.

[28] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization." in *Advances in Neural Information Processing Systems*, vol. 1, no. 1, 2007, pp. 2–1.

[29] ——, "Bayesian probabilistic matrix factorization using markov chain monte carlo," in *Proceedings of International Conference on Machine Learning*. ACM, 2008, pp. 880–887.

[30] P. Jain and I. Dhillon, "Provable inductive matrix completion," *arXiv preprint arXiv:1306.0626*, pp. 1–22, 2013. [Online]. Available: http://arxiv.org/abs/1306.0626

[31] K.-Y. Chiang, C.-J. Hsieh, and I. S. Dhillon, "Matrix completion with noisy side information," in *Advances in Neural Information Processing Systems*, 2015, pp. 3447–3455.

[32] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "Autorec: Autoencoders meet collaborative filtering," in *Proceedings of the 24th International Conference on World Wide Web*. ACM, 2015, pp. 111–112.

[33] R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted boltzmann machines for collaborative filtering," in *Proceedings of the 24th International Conference on Machine learning*. ACM, 2007, pp. 791–798.

[34] C. E. Seminario and D. C. Wilson, "Robustness and accuracy tradeoffs for recommender systems under attack." in *FLAIRS Conference*, vol. 314, 2012.

[35] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *International Conference on Learning Representations*. Citeseer, 2014.

[36] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations*, 2015.

[37] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2016, pp. 372–387.

[38] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *IEEE Symposium on Security and Privacy (SP)*. IEEE, 2016, pp. 582–597.

[39] C. Gong, D. Tao, W. Liu, L. Liu, and J. Yang, "Label propagation via teaching-to-learn and learning-to-teach," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 6, pp. 1452–1465, 2017.

[40] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 39–57.

[41] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *International Conference on Machine Learning*, 2018, pp. 274–283.

[42] Y. Bengio *et al.*, "Learning deep architectures for ai," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[43] C. Gülçehre and Y. Bengio, "Knowledge matters: Importance of prior information for optimization," *Journal of Machine Learning Research*, vol. 17, no. 1, pp. 226–257, 2016.

[44] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals, and Systems (MCSS)*, vol. 2, no. 4, pp. 303–314, 1989.

[45] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[46] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: a system for large-scale machine learning," in *Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation*. USENIX Association, 2016, pp. 265–283.

[47] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," in *Proceedings of the Twenty-fifth Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2009, pp. 452–461.

[48] X. He, T. Chen, M.-Y. Kan, and X. Chen, "Trirank: Review-aware explainable recommendation by modeling aspects," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 2015, pp. 1661–1670.