A Black-Box Attack Model for Visually-Aware Recommender Systems

Rami Cohen
rami_cohen@intuit.com
Intuit AI and Bar-Ilan University

Oren Sar Shalom
oren.sarshalom@gmail.com
Facebook

Dietmar Jannach
dietmar.jannach@aau.at
University of Klagenfurt

Amihood Amir
amir@esc.biu.ac.il
Bar-Ilan University

Abstract

Due to the advances in deep learning, visually-aware recommender systems (RS) have recently attracted increased research interest. Such systems combine collaborative signals with images, usually represented as feature vectors outputted by pre-trained image models. Since item catalogs can be huge, recommendation service providers often rely on images that are supplied by the item providers. In this work, we show that relying on such external sources can make an RS vulnerable to attacks, where the goal of the attacker is to unfairly promote certain *pushed items*. Specifically, we demonstrate how a new *visual attack* model can effectively influence the item scores and rankings in a black-box approach, i.e., without knowing the parameters of the model. The main underlying idea is to systematically create small human-imperceptible perturbations of the pushed item image and to devise appropriate gradient approximation methods to incrementally raise the pushed item's score. Experimental evaluations on two datasets show that the novel attack model is effective even when the contribution of the visual features to the overall performance of the recommender system is modest.

Keywords

Recommender Systems, Attacks, Adversarial Examples

ACM Reference Format:
Rami Cohen, Oren Sar Shalom, Dietmar Jannach, and Amihood Amir. 2021. A Black-Box Attack Model for Visually-Aware Recommender Systems. In *ACM WSDM '21.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/1122445.1122456

1 Introduction

During the last decades, with the increasing importance of the Web, Recommender Systems (RS) have gradually taken a more significant place in our lives. Such systems are used in various domains, from e-commerce to content consumption, where they provide value both for consumers and recommendation service providers. From a consumer perspective, RS for example help consumers deal with information overload. At the same time, recommender systems can create various types of economic value, e.g., in the form of increased sales or customer retention [15, 26, 27, 39, 43].

The economic value associated RS makes them a natural target of *attacks*. The goal of attackers usually is to *push* certain items from which they have an economic benefit, assuming that items that are ranked higher in the recommendation list are seen or purchased more often. Various types of attack models for RS have been proposed in the literature [18, 45]. The large majority of these attack models is based on injecting fake profiles into the RS. These profiles are designed in a way that they are able to mislead the RS to recommend certain items with a higher probability.

Nowadays, many deployed systems are however not purely collaborative anymore and do not base their recommendations solely on user preference profiles. Instead, oftentimes hybrid approaches are used that combine collaborative signals (e.g., explicit ratings or implicit feedback) with side information about the items. In the most recent years, it was moreover found that item images, i.e., the visual appearance of items, can represent an important piece of information that can be leveraged in the recommendation process. These developments led to the class of *visually-aware* recommender systems. Fueled by the advances in deep learning, a myriad of such approaches were proposed in recent years [2, 3, 11, 21, 22, 25, 29, 33].

In many domains, catalog sizes can be in the millions. Hence, from a practical point of view, it can be challenging or impractical for recommendation service providers, e.g., an e-commerce shop or media streaming site, to capture and maintain item images (e.g., product photos) for the entire catalog by themselves. Therefore, a common practice is to rely on the *item providers* to make their own images available to recommendation service providers by a dedicated API. This is a cross-domain practice, where e-commerce companies like Amazon[1] enable sellers to upload photos and content platform like YouTube[2] allow creators to upload their thumbnails. With the term *item providers*, we here refer to those entities (individuals or organizations) who act as suppliers of the items that are eventually recommended and who benefit economically from their items being listed in recommendations.

In modern visually-aware recommender systems, the features of the uploaded item photos, as discussed above, can influence the ranking of the items. Therefore, the item provider might be an *attacker* and try to manipulate the images in a way that the

[1]https://vendorcentral.amazon.com/
[2]https://support.google.com/youtube/answer/72431

underlying machine learning model ranks the item to be pushed higher in the recommendation lists of users. We call this novel way of manipulating a recommender system a *visual attack*.

While the literature on adversarial examples is rich, only few works exist on their use in the context of RS [13, 50]. Moreover, these works rely on impractical assumptions and require the attacker to have access to the internal representation of the RS. In contrast, the proposed algorithm assumes milder and more realistic assumptions. The main idea of the approach is to systematically manipulate the provided item images in a human-imperceptible way so that the algorithm used by the recommender predicts a higher relevance score for the item itself. Technically, we accomplish this by devising appropriate gradient approximation techniques. The contributions of our work are as follows:

- We identify a new vulnerability of RS, which emerges in situations where item providers are in control of the images that are used in visually-aware recommenders;
- We propose novel technical attack models that exploit this vulnerability under different levels of resources available to the adversary;
- We evaluate these models on two datasets, showing that they are effective even in cases where the visual component only plays a minor role for the ranking process;

To make our results reproducible, we share all data and code used in our experiments online[3].

2 Background and Related Work

We review three prerequisite research areas: visually-aware RS, attack models, and the use of adversarial examples in image classification.

2.1 Architecture of Visually-Aware RS

With respect to their technical architecture, state-of-the-art visually-aware recommender systems commonly incorporate high-level features of item images that were previously extracted from fixed pre-trained deep models. The system therefore invokes a pre-trained Image Classification model (IC), e.g., ResNet [20] or VGG [46], which returns the output of the penultimate layer of the model, dubbed the *feature vector* of the image. The advantage of using such pre-trained IC model is that they were trained on millions of images, allowing them to extract feature vectors that hold essential information regarding the visualization of the image.

However, the feature vector is *not* optimized for generating relevant recommendations. It is thus left to the recommender to learn a transformation layer that extracts valuable information with respect to recommendations. Usually this transformation is accomplished by multiplying the feature vector by a learned matrix E. Since the image is passed through a deterministic transformation, it defines a revised input layer having the feature vector replacing the pixels. A generic schematic visualization of this approach is depicted in Figure 1.

[3]https://github.com/vis-rs-attack/code

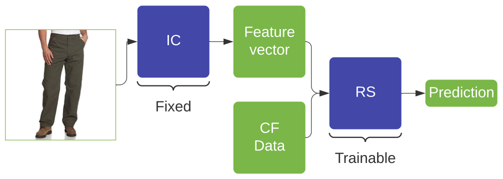

Figure 1: Common Architecture of a Visually-aware RS

VBPR [22] was probably the first work to follow this approach and extends BPR [40] to incorporate images. It maintains two embedding spaces, derived from the usage patterns and the item visualization. The final predicted score results from the sum of the scores of these two spaces. Sherlock [21] adds the ability to model the feature vector with respect to the item category, by learning a dedicated transformation matrix E_i per each category i. Later, DeepStyle [34] improved the category-aware approach by learning a single embedding matrix which maps the categories to the latent space of the images. Thus, a single transformation matrix E is sufficient. The authors of this paper showed that the reduced number of parameters leads to enhanced performance.

2.2 Attacks on Recommender Systems

The best explored attack models in the literature are based on the injection of fake users into the system, called *shilling*. These fake users leave carefully designed feedback that is intended to both seem genuine and to disrupt the output of the RS such that the pushed items will be recommended to the target users [18, 45]. This shilling attack can be highly effective for cold items, a ubiquitous phenomenon in real-life datasets [41].

The effectiveness of an attack is usually measured by the amount of target users that get the pushed item as a recommendation. In the case of shilling attacks, there is a positive correlation between effectiveness and adversary resources, where resources are usually composed of knowledge and capacity. The relevant knowledge includes several aspects, for example, knowing the type of the deployed model, observing the values of the parameter set, or even having read access to the rating dataset that empowers the model. The capacity refers to the amount of shilling users and fake feedback the adversary can inject. Trivially, with more knowledge and capacity the effectiveness of the attacks increases [38]. As we will show, the trade-off between resources and effectiveness also exists in visual attacks. However, the relevant resources in these types of attacks are entirely different from shilling attacks.

There are only very few existing works that rely on image information to attack an RS, as mentioned above, [13] and [50]. However, these works assume that the attacker can either manipulate the *internal representation* of the images as used by the RS, or has complete knowledge about the system. Moreover, in the case of [13], the goal is to change the predicted category classification of an item, whereas in our approach we aim to push items to the top-k lists of target users.

2.3 Adversarial Examples in Image Classification

Adversarial examples in machine learning are inputs that seem almost identical to examples from the true distribution of data that the algorithm works on. However, they are designed to deceive the model and cause it to output incorrect predictions. In the context of image classification, adversarial examples are constructed by adding imperceptible changes to genuine images, with the aim of crossing the decision boundary of the classifier (e.g., [49]). Adversarial examples can be generated either through *white-box* (WB) or *black-box* (BB) approaches.

White-box: This attack assumes the adversary has complete knowledge of the model and can compute gradients using backpropagation. Unlike regular stochastic gradient descent (SGD), which updates the model *parameters* in the *opposite* direction of the gradient, the adversary updates the *input* in the direction of the gradient. Let η be the adversarial direction. Then an adversarial example is generated as $\tilde{x} = x + \epsilon\eta$.

Black-box: Such an attack requires a milder assumption: the ability to query the model [7]. In the absence of gradients to guide the adversary, it invokes the model multiple times with various perturbations within a small volume around a target image and uses the associated feedback to infer the right direction. However, due to the high-dimensionality of the *pixel space*, even efficient attacks need at least tens of thousands of requests to the classifier, limiting their applicability [8, 19, 24, 52]. Our proposed approach does not operate on the pixel space, which in turn limits the number of queries to the model.

To assure the updated image does not deviate too much from the original one, some metric $d(\tilde{x}, x)$ is usually defined to quantify the visual dissimilarity attributed to the adversarial change. Commonly, d is computed by either ℓ_2 or ℓ_∞, and the goal is to deceive the model while keeping a low dissimilarity score to keep the change undistinguishable by humans. Various approaches were proposed to control the degree of visual dissimilarity. Examples include the "fast gradient sign method" [16], optimized for L_∞, which approximates this search problem by a binarization of the gradient, computed as $\eta \leftarrow sign(\eta)$. Early stopping was suggested by [31] and Madry et al. [35] used the "projected gradient descent" to limit the ℓ_2-norm.

Both white-box and black-box attacks can improve effectiveness by working iteratively and taking multiple small *steps* in the direction of the gradient. For instance, the "Iterative Gradient Sign" technique [30] applies multiple steps of the "fast gradient sign method" [16] and reports boost in performance.

3 Proposed Technical Approach

In this section we demonstrate how an attacker can exploit the identified vulnerability in visually-aware RS that are based on an architecture as sketched in Figure 1. We will first elaborate how a white-box attack can be mounted and then move on to more realistic black-box attacks. In particular we will show how to attack both predicted scores and rankings.

3.1 Preliminaries and Assumptions

The proposed visual attacks solely modify the visualizations of the pushed items. Therefore, pushing a certain item would not change the score of any other item. As a corollary, gradient computations in attacks with several pushed items are independent of each other. Hence, in this paper we assume there is a single pushed item.

There are three types of target users: specific users, segments or general population. Focusing on specific users is beneficial as it is equivalent to personal advertising and may exploit the influence of opinion leaders. However, in order to attack a specific user, the adversary has to know the consumption history of that user. Then, that user can be cloned by introducing a new user to the RS with the exact same usage patterns. The attack is carried out on the cloned user and the target user will, as a result, be affected by the attack.

Approaching general population dispenses with the need to have prior knowledge about individual users, but it may result in huge exposure of the pushed items to irrelevant users. A segmented attack is an in-between approach, where the adversary targets a set of users with similar tastes. As we will explain later, even in the general population and segmented attacks we can safely assume that the adversary attacks a *single* mock user that was *injected by the attacker* to the system.

3.2 An Upper Bound for Visual Dissimilarity

In the presented attack models, the item images are systematically perturbed in a way that the related item's score or ranking is improved. This process is done step-wise and at each step more noise is added to the image. In order to prevent the attack from being detected and to ensure good image quality, it is important to limit the number of steps.

To assess the perceptual visual distortion caused by the adversarial changes, we use the widely adopted *objective measure* Structural SIMilarity (SSIM) index [55]. This index considers image degradation as perceived change in structural information. The idea behind structural information is that pixels have stronger dependencies when they are spatially close. The SSIM index also incorporates, with equal importance, luminance masking and contrast masking terms. These terms reflect the phenomena where image distortions are more visible in bright areas and in the background, respectively.

The average SSIM indices after 10, 20 and 30 steps are: 0.965, 0.942 and 0.914, respectively (higher is better), which entails a noticeable degradation after 20 steps. Most images have a solid white background, however beyond 20 steps the attack might be revealed as some noise is noticed mostly in the background, which shows some unnatural texture. Therefore we set the maximal allowed number of steps to 20 in our experiments.

3.3 A White-Box Attack

To define an upper bound for the effectiveness of visual attacks as investigated in our paper, we show how a white-box attack can be designed. This attack assumes the adversary has a *read* access to the parameters of the model, which is in general not realistic.

At inference time, to allow personalized ranking, the score of user u for item i is computed by a visually-aware recommender as $s_{u,i} = f(u, i, p_i)$, where p_i is the image associated with item i. Note that the only assumption we make on $f(\cdot)$ is that it is differentiable. To avoid clutter, we omit the subscripts whenever their existence is clear from the context. To manipulate the image, the adversary computes the partial derivatives $\frac{\partial s}{\partial p}$ and updates the pixels in that direction, while keeping the distorted image as close as possible to the genuine one. Following [16], we binarize these partial derivatives by taking their sign, as our preliminary experiments asserted it leads to faster convergence. We note that any future improvements in adversarial examples for images may contribute to the effectiveness of this proposed approach.

The visual attack, as mentioned, is performed in small *steps*. Each step $t+1$ aims to increase the predicted score of the pushed item by updating the image as follows:

$$p^{t+1} = p^t + \epsilon \cdot sign(\frac{\partial s}{\partial p^t}) \tag{1}$$

where p^0 is the genuine image and ϵ is a small constant that controls the step size so the adversarial changes will be overlooked by the human eye.

3.4 A Black-Box Attack on Scores

As in the white-box attack, the goal of the adversary is to compute the gradient of the score s with respect to all pixels in the image p in the corresponding image, i.e., $\frac{\partial s}{\partial p}$. However, as explained in Section 2.3, black-box attacks in image classification usually require an enormous amount of requests to the classifier because they work in the *pixel space*. Applying similar techniques in visual attacks would therefore be impractical.

Remember that the RS subnetwork is unknown to the attacker (neither its parameters or even its architecture). However, remember that in our target architecture for visually-aware RS (Figure 1), we assume that the system leverages a pretrained IC model. As discussed in Section 2.1, such IC models are publicly available and held constant during training, while the rest of the network is optimized for the specific task and data. In our case, given a new image, the RS extracts a feature vector f_i from image p_i by feed forwarding it through the IC and outputting the penultimate layer.

This situation now allows us to apply the chain rule and the multiplication between the partial derivatives of the score s with respect to f and the Jacobian of f with respect to p can be computed. Formally: $\frac{\partial s}{\partial p} = \frac{\partial s}{\partial f} \cdot \frac{\partial f}{\partial p}$. Upon computing this value, a step is made as defined in Eq. 1. Since we assume the parameter set of the image processing unit is known, $\frac{\partial f}{\partial p}$ can be analytically computed using backpropagation.

3.5 Computation of $\frac{\partial s}{\partial f}$

Decoupling the partial derivative into two quantities allows to avert numerical computations in the pixel space. However, the computation of the partial derivatives $\frac{\partial s}{\partial f}$ under the black-box model of the RS is rather complicated because f is not the actual input layer.

To illustrate the challenge, consider the following approach for numerical computation of the partial derivatives, where d denotes the dimensionality of f. Create d perturbations of p, such that each perturbation p^k, for $0 \leq k < d$, under the constraint $\left\|p - p^k\right\| \leq \delta$, yields a feature vector $f^k = f + \epsilon \cdot \mathcal{I}(k)$ for some predefined δ and arbitrary small ϵ, where $\mathcal{I}(k)$ is the one-hot vector with 1 for k-th coordinate and 0 elsewhere. That is, each perturbation isolates a different dimension in f. Then, replace the genuine image d times, each time with a different perturbation, and obtain from the RS a new personalized score s^k. Now the numerical partial derivative $\frac{\partial s}{\partial f^k}$ is given by $\frac{s^k - s}{\epsilon}$. However, since each pixel in p holds a discrete value and affects all dimensions in f, finding such a set of perturbed images is intractable.

Furthermore, in the absence of the ability to directly control f, it is not clear how to apply existing methods of gradient estimation. For instance, the widely used NES [56] requires to sample perturbations from a Gaussian distribution centered around the input f; even the seminal SPSA algorithm [47] demands a *symmetric* set of perturbations. As such methods are unattainable, we devise a novel approach to estimate the gradients. This method does not have any constraints on the structure of the perturbations. Moreover, the number of perturbations required by this method is *sublinear* in d. To this end, we first present a method that requires exactly d perturbations and then show how this amount can be reduced. The attacker creates d random perturbations of the image uniformly drawn from the L_∞-ball centered around p with radius δ, for some arbitrary small δ, and obtains their corresponding feature vectors f^k together with the associated personalized scores s^k. Zou et al. [60] proved that CNN-based architectures are Lipschitz continuous, and empirically showed that their Lipschitz constant is bounded by 1. Consequently, since each perturbation is similar to the original image, up to a tiny difference δ, also the deltas in the feature vectors $\left\|f^k - f\right\|$ are expected to be epsilonic, making them suitable for numeric computation of $\frac{\partial s}{\partial f}$. Using matrix notation, the set of vectors $\{f^k\}$ and scalars $\{s^k\}$ are referred to as matrix $\boldsymbol{F} \in \mathbb{R}^{d \times d}$ and vector $\boldsymbol{s} \in \mathbb{R}^d$, correspondingly.

This translates into a linear system $\boldsymbol{F}x = \boldsymbol{s}$ such that its solution x is the numerical computation of $\frac{\partial s}{\partial f}$. Keep in mind that a linear system is shift invariant. That is, the same solution x is obtained upon subtracting f^T from each row of $\boldsymbol{F}$ and subtracting s from each component in $\boldsymbol{s}$. Now the coefficient matrix is the change in f and the dependent variable is the resulting change in score:

$$x = (\boldsymbol{F} - f^T)^{-1}(\boldsymbol{s} - s) = \Delta \boldsymbol{F}^{-1} \Delta \boldsymbol{s} = \frac{\partial s}{\partial f} \tag{2}$$

Note that a single solution exists if and only if $\boldsymbol{F}$ is nonsingular. Since it is obtained by random perturbations, it is almost surely the case [28]; otherwise a new set of perturbations can be drawn.

Reducing the Number of Perturbations Each perturbation entails uploading a new image to the server. In ResNet, for example, $d = 2,048$, which results in a large amount of updates. We next address this problem and suggest a trade-off between effectiveness of each step and amount of required image updates.

Note that although Eq. 2 requires d perturbations for finding a single solution, an approximation can be found with fewer perturbations. Formally, let d' be the number of perturbations the adversary generates, for some $d' < d$. This defines an underdetermined system of linear equations, and among the infinite solutions as the estimated gradients, the one with the minimal ℓ_2 norm can be found [9]. Another advantage of reducing the number of perturbations is the ability to generalize. There is an unlimited number of possible perturbations, and every subset of d perturbations defines another linear system, with another exact solution. Those exact solutions vary in their distances from the true, unobserved derivatives. However, since the adversary randomly select a single subset of perturbations, the exact solution may "overfit" to this instantiation of the linear system, and not generalize to other solutions that could be yielded from other subsets. Such behavior might be highly problematic, and must be addressed correspondingly. To remedy this problem, regularization comes in useful. Reducing the number of perturbations, and finding a solution with minimal ℓ_2 norm can be thought of as a means of regularization, and can potentially improve performance. Indeed, as we show in Section 4, our approach not only reduces the required amount of perturbations also leads to improved performance.

3.6 A Black-Box Attack on Rankings

The attack described in the previous section is suited to target RS that reveal the scores, e.g., in the form of predicted ratings. A more common situation, however, is that we can only observe the personalized rankings of the items. Here, we therefore describe a novel black-box method that operates on the basis of such rankings.

In this attack, the adversary generates d perturbations as well, and the scores s_k should be recovered by using only their ranking r_k. If the score distribution of the d perturbations was known, the inverse of the CDF could accurately recover the scores. A common assumption is that item scores follow normal distribution [48]. However, even if it is true in practice, its mean and variance remain unknown. As observed from Eq. 2, the solution is shift invariant, so the mean of the distribution is irrelevant, but the variance does play a role.

Proposed Approach To bypass this problem we note that perhaps the entire catalog's scores follow a Gaussian distribution, where various items are involved. However, the differences in the perturbations' scores stem only from small changes to the image, while the pushed item itself is fixed. Therefore, we assume the scores reside within a very small segment $[s_{min}, s_{max}]$ of the entire distribution. If the segment is small enough, it can be approximated by a uniform distribution. Hence, a mapping between ranking and score is given by $s_k = \frac{N-r_k}{N}(s_{max} - s_{min})$, where N is the number of items in the catalog.

The remaining question is how to compute these values without knowledge of s_{min} or s_{max}. First, as Eq. 2 is shift invariant, we can assume $s_{min} = 0$. Second, because we solve a linear system, multiplying the dependent variable (scores) by a scalar, also scales the solution (derivatives) by the same factor. Since we take the sign of the gradient, scaling has no effect on the outcome. Thereby, we can assume $s_{max} = 1$, and therefore $s_k = 1 - \frac{r_k}{N}$.

Remarkably, this simple approach performed well in our initial experiments. In these experiments, we created a *privileged baseline* that receives the empirical standard deviation as input (information that is not accessible to the adversary), which allows it to recover the scores using the inverse of the CDF of the appropriate normal distribution. Nevertheless, assuming uniform distribution outperformed that baseline. We explain this phenomenon by the fact that the perturbations do not perfectly follow a Gaussian distribution, which leads to inconsistency in the inferred scores.

Dealing with Partial Ranking Knowledge: One possible limitation of our algorithm is the assumption that the complete ranking of all items in the catalog is revealed by the RS. However, in some cases not all catalog items are shown even when the user scrolls down the recommendation list. Therefore, the pushed item might not be shown to the attacker, which averts the proposed attack since no gradients could be computed. To overcome this obstacle, a surrogate user u' is introduced by the attacker and set as the attacked one. u' maintains two properties: 1) receives the pushed item in the top of the list, so gradients can be computed; 2) is similar to u, so the computed gradients are useful for the attacker.

Specifically, u' is identical to the target user u, but has an added item i' in the consumption history. This item is carefully selected, such that it makes the pushed item penetrate to the top items of user u', thus allowing to compute $\frac{\partial s_{u',i}}{\partial f_i}$. Note that this partial derivative is close to $\frac{\partial s_{u,i}}{\partial f_i}$ but is not identical, as it is affected by i'. Let n denote the number of items in the history of u and $u(i)$ a user with only item i in the consumption history. Naturally the relative contribution of i' to the gradient decreases with n. As the attacker has no prior knowledge on the underlying RS, by simply assuming a linear correlation, the contribution of i' is counteracted as follows: $\frac{\partial s_{u,i}}{\partial f_i} = \frac{(n+1)\cdot\frac{\partial s_{u',i}}{\partial f_i} - \frac{\partial s_{u(i'),i}}{\partial f_i}}{n}$. Once i appears in the top of u's recommendations, the attack is performed directly on u. Item i' is selected from the top recommendations for $u(i)$. In most cases, adding item i' to the consumption history of u is not enough to place i in the top of the list of u'. Therefore, several attack steps are performed on $u(i')$, to increase $s_{u(i'),i}$ until i appears at the top of the list of u'. Overall, in order to push item i, the algorithm requires to create 2 additional users: $u(i)$ and $u(i')$, which is an attainable request in general.

4 Experimental Evaluation

We demonstrate the effectiveness of the attack models by conducting experiments on several combinations of RSs and datasets.

4.1 Experiment Setup

4.1.1 Datasets We relied on two real-world datasets that were derived from the Amazon product data [36]. The first dataset is the Clothing, Shoes and Jewelry dataset (named "Clothing" for short) and the second one is the Electronics dataset. These datasets differ in all major properties, like domain, sparsity and size, which shows visual attacks are a ubiquitous problem. Arguably chief among their distinctive traits is the importance of visual features, quantitatively approximated by the gain in performance attributed to the ability to model images. To compute this trait, dubbed as "visual gain", we use AUC [40] as a proxy for performance. Since VBPR [22] extends BPR [40] to model images, the visual gain is computed as the relative increase in the AUC obtained by VBPR over BPR.

Table 1 summarizes statistics of these datasets. It is noticed that the visual gain in the electronics dataset ($2\% = \frac{0.85}{0.83} - 1$) is much lower than in the clothing dataset ($23\% = \frac{0.79}{0.64} - 1$). We therefore expect that visual features in the electronics domain would be of lesser importance, and as a consequence the RS will only moderately consider them. Nevertheless, we show that visual attacks are effective even in this domain.

4.1.2 Attacked Models We experiment with two visually-aware RS as the underlying attacked models. The first is the seminal algorithm VBPR [22]. To assess the effect of visual attacks in a common setting where *structured* side information is available, the second model

Dataset	#users	#items	#feedback	Visual gain
Clothing	127,055	455,412	1,042,097	23%
Electronics	176,607	224,852	1,551,960	2%

Table 1: Dataset statistics after preprocessing

is DeepStyle [34], which also considers the item categories. We train a model for each combination of dataset and underlying RS, yielding 4 models in total. We randomly split each dataset into training/validation sets, in order to perform hyperparameter tuning. Both algorithms are very competitive and the obtained AUC of VBPR (DeepStyle) on the validation set of the Clothing dataset is 0.79 (0.80) and 0.85 (0.86) in the Electronics domain.

4.1.3 Visual Features The underlying image classification model is ResNet [20]. This is a 50 layer model, trained on 1.2 million ImageNet [12] images. For each item to be modeled by the RS, its associated image is first fed into the IC model and the penultimate layer of size 2,048 is extracted as the feature vector.

4.1.4 Evaluation Metric The purpose of attacks is to increase the visibility of the pushed items in top-k recommendations. Therefore, we follow [37, 42] and evaluate the effectiveness of an attack using the Hit-Ratio (HR), which measures the fraction of affected users by the attack. Formally, let $\mathcal{S}$ be the set of user-item pairs subject to an attack. For each $(u, i) \in \mathcal{S}$, the Boolean indicator $H^k_{u,i}$ is true if and only if item i is in the top-k recommendation to user u. Then the Hit Ratio is defined as follows:

$$HR@k = \frac{1}{|\mathcal{S}|} \sum_{(u,i)\in\mathcal{S}} H^k_{u,i}$$

In real-life datasets with many items, the probability of a random pushed item to appear in the top of the recommendation list is extremely low. Therefore HR directly measures the impact of the attack, as the number of pre-attack hits is negligible or even zero.

4.2 Varying the Target Population

We design dedicated experiments for each type of target populations to reflect the different assumed capabilities of the adversary.

4.2.1 Specific-users attack In this threat, the adversary knows the consumption list of the attacked user and hence can impersonate that user by introducing to the RS a new user with the same list. To simulate these attacks, we randomly select 100 user-item pairs to serve as the attacked users and pushed items.

4.2.2 Segmented attack Following [4], a segment is a group of users with similar tastes, approximated by the set of users with a common specific item in their interaction histories. For instance, a segment can be defined as all users who purchased a certain Adidas shoe. We name the item that defines the segment as the *segment item*[4]. To perform this attack, the adversary creates a single mock user who serves as the target user, whose interaction history includes only the segment item. We do not assume that the adversary possesses information about usage patterns in general or knows which item defines the optimal segment item. Therefore, the segment item is simply chosen by random among all items in the same category of the pushed item. In our experiments we chose 100 random pairs of pushed and target items. We emphasize that the actual target users are those who interacted with the segment item, but we do not assume the adversary knows their identities. We report the effectiveness of the attack on the actual set of target users, which is obscured from the adversary.

4.2.3 General population This attack assumes that the adversary wishes to associate the pushed item with every user in the dataset. As this threat model does not assume the adversary has any knowledge of usage behavior, preferences of mass population need to be approximated. To this end, an adversary might pick N random items, naturally without knowing which users have interacted with them. The assumption is if N is large enough, then these items span diverse preferences. Then the adversary adds these items by injecting N users, each with a single item in their interaction history, to function as the set of target users. The *active target* user is chosen in a round-robin manner, and an adversarial step is made to maximize the score for that user.

However, we noticed that this method fails, as maximizing the score for the active target user also deteriorates the scores of the pushed item for many other target users. This is an expected behavior, as it is infeasible to find a direction that increases the scores of an entire set of randomly selected users.

Keeping in mind that the objective of the adversary is not to increase the average score of the pushed item, but to bubble it up to the *top* of many users' lists, we modify this approach by dynamically choosing the next active user. Preceding to each step, the adversary ranks the target users by their personalized score of the pushed item[5]. Then it concentrates on target users who are more likely to receive the pushed item in the top of their recommendation list. This is done by setting the active target user as the one in the p^{th} percentile. As p increases from 0 to 100%, the adversary concentrates on a larger portion of the target users. we tuned p on 5 equally distanced values in the range $[0, 1]$ and obtained optimal performance at $p = 25\%$.

In this type of experiment we pick at random 100 pairs of items, each of which serves as the pushed item and set $N = 200$ as the number of random items. Once again, the adversary cannot know the actual effectiveness of this attack on the general population, since their identities and usage patterns are not disclosed. We report the performance of the attack on $1,000$ randomly chosen users who serve as a representative sample of the general population.

4.3 Baselines

Although the main purpose of this paper is to empirically demonstrate the existence of a threat on AI safety posed by visual attacks, it is still important to compare the effectiveness of our methods with relevant baselines. To empirically demonstrate the existence of the identified vulnerability and to judge the relative effectiveness of the proposed attack models, we include two baseline attacks in our experiments. Since, to the best of our knowledge, this is the first work to investigate these types of visual attacks in the context of recommenders, there are no prior baselines.

[4]Supporting multiple segment items can be trivially implemented as well.

[5]Note that this operation adds N parallelable queries to the recommender and does not require to alter the image.

Nevertheless, to show the importance of carefully designed perturbations, we propose two simple baseline methods that perform push attacks by altering genuine images.

- The first is an easy-to-detect attack, which replaces the original image p of the pushed item with the image p_{pop}, associated with the most popular item from the same category. The rationale behind this baseline is that some of the success of the most popular items is attributed to their images. A visually-aware recommender should capture the visual features that correlate with consumption, and reflect it in superior scores for items with these images.
- The second baseline is a softer version of the former, making it harder to be detected. In this baseline, $p \longleftarrow p + \epsilon \cdot p_{pop}$, where ϵ controls the amount of change added to the image. We experimented values in the range $[0.01, 0.1]$ in steps of 0.01.

However, while in all configurations both baselines systematically increased the scores of the pushed items and improved their ranking in thousands of places on average, they failed to penetrate to the top of the lists and to improve *HR@K*. The resulting Hit Ratios were therefore consistently very close to zero in all experiments, which is why we omit the results in the following sections. Ultimately, this evidence reinforces our preliminary assumption on the importance of well-guided adversarial examples.

4.4 Results

Here, we report the results of the experiments and present multiple trade-offs between adversarial resources and effectiveness. Across all experiments, when partial ranking knowledge is available, we assume that only the top 1% of the ranking is known.

4.4.1 Specific-user Attack Table 2 details the HR@k of different attacks for various values of k. We fixed the number of perturbations (32) and steps (20), leading to a total of $640 = 32 \cdot 20$ image updates. Attack types denoted by WB, BB-Score, BB-Rank and BB-Partial stand for white-box, black-box on scores, black-box on rankings and black-box on partial ranking knowledge, respectively.

The results demonstrate the effectiveness of all proposed attack models. In terms of the absolute numbers for the Hit Ratio we can observe that after the attack the probability of a randomly pushed item to appear in the top-20 list is substantial, even when using a black-box approach. Generally, the results also validate the existence of trade-off between knowledge on the attacked system and effectiveness. We notice that the white-box attack, which assumes absolute knowledge on the underlying system, as expected outperforms the rest of the attacks. However, the differences between the black-box approaches on rankings and scores are usually small and depend on the metric and dataset.

It is worthwhile to mention that even in the Electronics dataset, where visual features play a minor role, with attributed gain of only 2% to the AUC, the attack is still effective. For a considerable amount of users, it can bubble up random items to the top of their lists. This proves that under attack, even seemingly less important side-information can be used to manipulate the results of collaborative filtering models.

Dataset	RS	Attack	*HR@1*	*HR@10*	*HR@20*
Clothing	VBPR	WB	0.79	0.89	0.91
		BB-Score	0.43	0.54	0.58
		BB-Rank	0.41	0.51	0.53
		BB-Partial	0.23	0.23	0.23
	DeepStyle	WB	0.73	0.80	0.83
		BB-Score	0.39	0.50	0.52
		BB-Rank	0.42	0.50	0.54
		BB-Partial	0.10	0.10	0.10
Electronics	VBPR	WB	0.45	0.57	0.60
		BB-Score	0.15	0.22	0.27
		BB-Rank	0.14	0.24	0.27
		BB-Partial	0.47	0.50	0.50
	DeepStyle	WB	0.53	0.62	0.66
		BB-Score	0.18	0.20	0.29
		BB-Rank	0.20	0.26	0.29
		BB-Partial	0.20	0.20	0.20

Table 2: Efficacy of specific-user attack

Due to space limitations, in the rest of the paper we report the results only of a subset of the configurations. First, as we did not find any significant difference in terms of the attack effects for the different algorithms (VBPR and DeepStyle) we focus on the former. Second, while the HR of Clothing and Electronics are different, their trends across the experiments remain the same. Hence, we report the results of the Clothing dataset. Third, the default value of k is 20 in the *HR@k* measure. Moreover, in the experimented datasets across the three color channels, pixels take values in the range $(0, 255)$. We report the results obtained by setting the minimal step size ϵ, which bounds the size of each update to at most 1, as it led to superior performance comparing to higher values.

Figures 2 and 3 extend Table 2 and probe the effects of isolated factors. Figure 2 visualizes the trade-offs between effectiveness, number of steps, and number of perturbations under the BB-Rank attack. It plots HR as a function of the number of steps, for several values of perturbations per step.

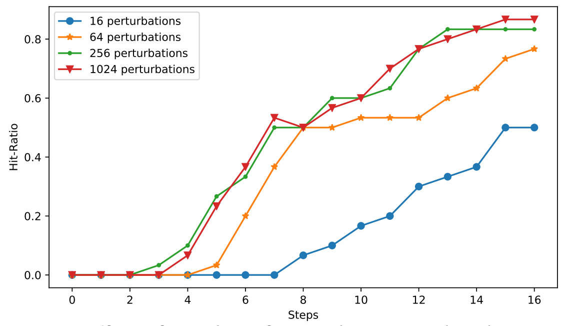

Figure 2: Effect of number of perturbations and update steps

As can be seen, if the adversary employs 64 perturbations per step and wants to obtain HR of roughly 0.5, then 8 steps are required, which means a total of $512 = 64 \cdot 8$ image updates. In comparison, obtaining on-par performance with 16 perturbations requires to almost double the number of steps, and to take 15 steps, which might deteriorate the visualization of the image. However, that

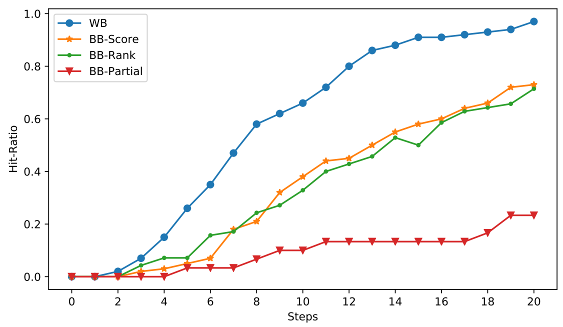

Figure 3: Effectiveness of each attack over number of steps

alternative requires only $240 = 16 \cdot 15$ image updates. We note that using 2,048 perturbations leads to a single solution of the gradient, without ability to generalize. Its effect is detrimental and these attacks fail to increase the HR.

In the rest of the experiments we fix the number of perturbations to 64, as it achieves solid performance with a moderate budget of image updates. Figure 3 shows the effectiveness of each attack at various numbers of steps. We first observe that the effectiveness of the four types of attacks monotonically increases with the number of steps. This demonstrates the effectiveness of our computations, as each additional step directs the adversary to achieve more impact. The BB-Score and BB-Rank behave very similarly throughout the attack, which shows our proposed method of estimating scores by ranking is consistent. Ranking based on partial knowledge also leads to solid performance, which shows the potential of this attack in real-life scenarios.

Finally, we hypothesize that the effectiveness of an attack depends on the relevance of the pushed item to the target user, approximated by the pre-attack ranking. Figure 4 shows the effectiveness of the BB-Rank attack where the pushed items are grouped by their initial ranking. As it shows, attacks on relevant items converge faster and achieve higher HR. Remarkably, even when the pushed item is irrelevant to the user, i.e., the initial ranking is in the hundreds of thousands, within a plausible number of 20 steps, the adversary succeeds in pushing the item to the top of the lists of the vast majority of the target users.

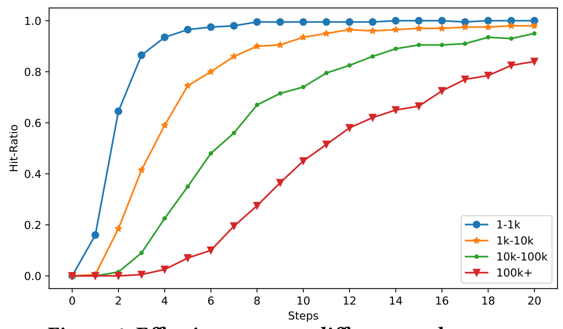

Figure 4: Effectiveness over different rank ranges

4.4.2 Segmented and General Population Attacks Figure 5 shows the performance of the segmented and general population attacks side by side. Again, we can observe a correlation between the adversary's knowledge about the target users and performance. Compared to the general population attack, the Hit Ratio is higher for the segmented attack, where the adversary knows a single item in the history of all users (although does not know the identity of them). Furthermore, both of these attacks are outperformed by the specific-user attacks (not shown in the figure), in which the adversary possesses vast knowledge on the interaction history of the users. This comes as no surprise, since deep knowledge on users entails more accurate guidance for the adversary. When given only *partial* ranking knowledge, the attack is still successful for a significant amount of users, e.g., around 1.6% of the entire user base for the general population attack.

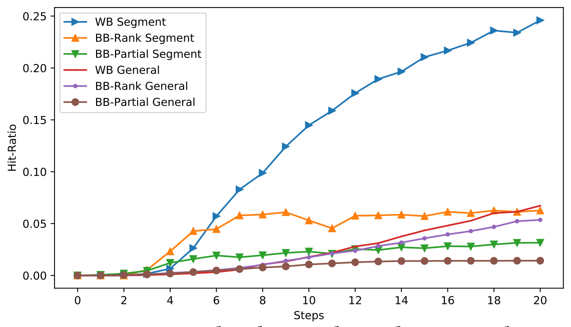

Figure 5: Segmented and general population attacks

5 Discussion and Outlook

In this paper we identified and investigated a new type of vulnerability of recommender systems that results from the use of externally-provided images. We devised different white-box and, more importantly, black-box attack models to exploit the vulnerabilities. Experiments on two datasets and showed that the proposed attack models are effective in terms of manipulating the scores or rankings of two visually-aware RS.

While the proposed approaches rely on images to perform attacks, the same techniques can be applied on any continuous side information. Consequently, a possible future work is to investigate the vulnerabilities of audio-based [54], video-based [10] or textual review-based [44] recommenders .

We hope this paper will raise awareness about image-based attacks and spur research on means to combat such attacks. In the context of defending RS against shilling attacks, we are encouraged by the impressive effectiveness of detection algorithms [5, 32, 51, 57, 59], and the results of robust RS [1, 53]. Also, existing works on increasing the robustness of RS against adversarial examples [14, 23] may ignite research on robustness against visual-attacks. Additionally, research against adversarial examples in general is fruitful and several algorithms were proposed for detection [6, 58] and new robust models were devised [5, 17]. We anticipate a plethora of work in these veins towards more robust recommenders, which are more robust to tiny visual changes.

References

[1] Santiago Alonso, Jesús Bobadilla, Fernando Ortega, and Ricardo Moya. 2019. Robust model-based reliability approach to tackle shilling attacks in collaborative filtering recommender systems. *IEEE Access* 7 (2019), 41782–41798.

[2] Svetlin Bostandjiev, John O'Donovan, and Tobias Höllerer. 2012. TasteWeights: a visual interactive hybrid recommender system. In *RecSys '12*. 35–42.

[3] Simon Bruns, André Calero Valdez, Christoph Greven, Martina Ziefle, and Ulrik Schroeder. 2015. What should I read next? A personalized visual publication recommender system. In *HIMI 2015*. 89–100.

[4] Robin Burke, Bamshad Mobasher, Runa Bhaumik, and Chad Williams. 2005. Segment-based injection attacks against collaborative filtering recommender systems. In *ICDM'05*. 577–580.

[5] Robin Burke, Bamshad Mobasher, Chad Williams, and Runa Bhaumik. 2006. Classification features for attack detection in collaborative recommender systems. In *KDD '06*. 542–547.

[6] Nicholas Carlini and David Wagner. 2017. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. 3–14.

[7] Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. 2018. Adversarial attacks and defences: A survey. *arXiv preprint arXiv:1810.00069* (2018).

[8] Minhao Cheng, Thong Le, Pin-Yu Chen, Jinfeng Yi, Huan Zhang, and Cho-Jui Hsieh. 2018. Query-efficient hard-label black-box attack: An optimization-based approach. *arXiv preprint arXiv:1807.04457* (2018).

[9] Randall E. Cline and Robert J. Plemmons. 1976. l_2-Solutions to Underdetermined Linear Systems. *SIAM Rev.* 18, 1 (1976), 92–106.

[10] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *RecSys '16*. 191–198.

[11] Yashar Deldjoo, Mehdi Elahi, Paolo Cremonesi, Franca Garzotto, Pietro Piazzolla, and Massimo Quadrana. 2016. Content-based video recommendation system based on stylistic visual features. *Journal on Data Semantics* 5, 2 (2016), 99–113.

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *CVPR '09*. 248–255.

[13] Tommaso Di Noia, Daniele Malitesta, and Felice Antonio Merra. 2020. TAaMR: Targeted Adversarial Attack against Multimedia Recommender Systems. In *the 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-DSML'20)*.

[14] 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.

[15] Daniel Fleder and Kartik Hosanagar. 2009. Blockbuster culture's next rise or fall: The impact of recommender systems on sales diversity. *Manag. Sci.* 55, 5 (2009).

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

[17] Shixiang Gu and Luca Rigazio. 2014. Towards deep neural network architectures robust to adversarial examples. *arXiv preprint arXiv:1412.5068* (2014).

[18] Ihsan Gunes, Cihan Kaleli, Alper Bilge, and Huseyin Polat. 2014. Shilling attacks against recommender systems: a comprehensive survey. *Artificial Intelligence Review* 42, 4 (2014), 767–799.

[19] Chuan Guo, Jacob R Gardner, Yurong You, Andrew Gordon Wilson, and Kilian Q Weinberger. 2019. Simple black-box adversarial attacks. *arXiv preprint arXiv:1905.07121* (2019).

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR '16*. 770–778.

[21] Ruining He, Chunbin Lin, Jianguo Wang, and Julian McAuley. 2016. Sherlock: sparse hierarchical embeddings for visually-aware one-class collaborative filtering. *arXiv preprint arXiv:1604.05813* (2016).

[22] Ruining He and Julian McAuley. 2016. VBPR: visual bayesian personalized ranking from implicit feedback. In *AAAI '16*.

[23] Xiangnan He, Zhankui He, Xiaoyu Du, and Tat-Seng Chua. 2018. Adversarial personalized ranking for recommendation. In *SIGIR '16*. 355–364.

[24] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. 2018. Black-box adversarial attacks with limited queries and information. *arXiv preprint arXiv:1804.08598* (2018).

[25] Tomoharu Iwata, Shinji Watanabe, and Hiroshi Sawada. 2011. Fashion coordinates recommender system using photographs from fashion magazines. In *IJCAI '11*.

[26] Dietmar Jannach and Michael Jugovac. 2019. Measuring the Business Value of Recommender Systems. *ACM Trans. on Mgt. Inf. Sys.* 10, 4 (2019).

[27] Dietmar Jannach, Oren Sar Shalom, and Joseph A Konstan. 2019. Towards more impactful recommender systems research. In *Proceedings of the ACM RecSys Workshop on the Impact of Recommender Systems (ImpactRS'19)*.

[28] Jeff Kahn, János Komlós, and Endre Szemerédi. 1995. On the probability that a random±1-matrix is singular. *Journal of the American Mathematical Society* 8, 1 (1995), 223–240.

[29] Chan Young Kim, Jae Kyu Lee, Yoon Ho Cho, and Deok Hwan Kim. 2004. Viscors: A visual-content recommender for the mobile web. *IEEE Intelligent Systems* 19, 6 (2004), 32–39.

[30] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2016. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533* (2016).

[31] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2016. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236* (2016).

[32] Wentao Li, Min Gao, Hua Li, Jun Zeng, Qingyu Xiong, and Sachio Hirokawa. 2016. Shilling attack detection in recommender systems via selecting patterns analysis. *IEICE Transactions on Information and Systems* 99, 10 (2016), 2600–2611.

[33] Maria Teresa Linaza, Amaia Agirregoikoa, Ander Garcia, Jose Ignacio Torres, and Kepa Aranburu. 2011. Image-based travel recommender system for small tourist destinations. In *ENTER '11*. 1–12.

[34] Qiang Liu, Shu Wu, and Liang Wang. 2017. DeepStyle: Learning user preferences for visual recommendation. In *SIGIR '17*. 841–844.

[35] 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).

[36] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *SIGIR '15*. 43–52.

[37] Bhaskar Mehta and Wolfgang Nejdl. 2008. Attack resistant collaborative filtering. In *SIGIR '08*. 75–82.

[38] Bamshad Mobasher, Robin Burke, Runa Bhaumik, and Jeff J Sandvig. 2007. Attacks and remedies in collaborative recommendation. *IEEE Intelligent Systems* 22, 3 (2007), 56–63.

[39] Bhavik Pathak, Robert Garfinkel, Ram D Gopal, Rajkumar Venkatesan, and Fang Yin. 2010. Empirical analysis of the impact of recommender systems on sales. *Journal of Management Information Systems* 27, 2 (2010), 159–188.

[40] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian personalized ranking from implicit feedback. *UAI '09* (2012).

[41] Oren Sar Shalom, Shlomo Berkovsky, Royi Ronen, Elad Ziklik, and Amir Amihood. 2015. Data quality matters in recommender systems. In *Proceedings of the 9th ACM Conference on Recommender Systems*. 257–260.

[42] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW '01*. 285–295.

[43] J Ben Schafer, Joseph Konstan, and John Riedl. 1999. Recommender systems in e-commerce. In *EC '99*. 158–166.

[44] Oren Sar Shalom, Guy Uziel, and Amir Kantor. 2019. A generative model for review-based recommendations. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 353–357.

[45] Mingdan Si and Qingshan Li. 2020. Shilling attacks against collaborative recommender systems: a review. *Artificial Intelligence Review* 53, 1 (2020), 291–319.

[46] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[47] James C Spall. 1992. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Trans. Automat. Control* (1992).

[48] Harald Steck. 2015. Gaussian ranking by matrix factorization. In *RecSys '15*.

[49] 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).

[50] Jinhui Tang, Xiaoyu Du, Xiangnan He, Fajie Yuan, Qi Tian, and Tat-Seng Chua. 2019. Adversarial training towards robust multimedia recommender system. *IEEE Transactions on Knowledge and Data Engineering* (2019).

[51] Chao Tong, Xiang Yin, Jun Li, Tongyu Zhu, Renli Lv, Liang Sun, and Joel JPC Rodrigues. 2018. A shilling attack detector based on convolutional neural network for collaborative recommender system in social aware network. *Comput. J.* (2018).

[52] Chun-Chen Tu, Paishun Ting, Pin-Yu Chen, Sijia Liu, Huan Zhang, Jinfeng Yi, Cho-Jui Hsieh, and Shin-Ming Cheng. 2019. Autozoom: Autoencoder-based zeroth order optimization method for attacking black-box neural networks. In *AAAI '19*, Vol. 33. 742–749.

[53] Ahmet Murat Turk and Alper Bilge. 2018. A Robust Multi-Criteria Collaborative Filtering Algorithm. In *INISTA '18*. 1–6.

[54] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. 2013. Deep content-based music recommendation. In *NIPS*.

[55] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing* 13, 4 (2004), 600–612.

[56] Daan Wierstra, Tom Schaul, Jan Peters, and Juergen Schmidhuber. 2008. Natural evolution strategies. In *CEC '08*. 3381–3387.

[57] Chad A Williams, Bamshad Mobasher, and Robin Burke. 2007. Defending recommender systems: detection of profile injection attacks. *Service Oriented Computing and Applications* 1, 3 (2007), 157–170.

[58] Weilin Xu, David Evans, and Yanjun Qi. [n.d.]. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155*.

[59] Fuzhi Zhang, Zening Zhang, Peng Zhang, and Shilei Wang. 2018. UD-HMM: An unsupervised method for shilling attack detection based on hidden Markov model and hierarchical clustering. *Knowledge-Based Systems* 148 (2018), 146–166.

[60] Dongmian Zou, Radu Balan, and Maneesh Singh. 2019. On Lipschitz Bounds of General Convolutional Neural Networks. *IEEE Trans. on Information Theory* (2019).