UNIVERSITY OF CALIFORNIA
RIVERSIDE

# Defending Against Adversarial Attacks With Low-Rank Factorization

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

by

Negin Entezari

March 2022

Dissertation Committee:

Dr. Evangelos Papalexakis, Chairperson
Dr. Vassilis Tsotras
Dr. Eamonn Keogh
Dr. Michalis Faloutsos

## Acknowledgments

I would like to thank people who helped me along my PhD journey and without whom I would not have been able to complete this research and dissertation. First and foremost, I am extremely grateful to my advisor, Dr. Vagelis Papalexakis, for his invaluable advice and endless support during my PhD study. He has inspired me to pursue my academic and career goals. It was an honor and a pleasure to work with him. I would also like to thank my committee members, Dr. Vassilis Tsotras, Dr. Eamonn Keogh, and Dr. Michalis Faloutsos, for their insightful and valuable comments and suggestions. I am especially grateful to Dr. Tsotras who has been the greatest and the most supportive graduate advisor throughout my PhD study at UCR.

I am also thankful to my lab mates and friends at Multi-Aspect Data lab, Sara Abdali, Yorgos Tsitsikas, Ekta Gujral, Ravdeep Pasricha, Rutuja Gurav, and Uday Singh Saini, for their friendship and help along the journey. I would also like to thank all my friends at UCR for being part of this wonderful journey.

I am extremely grateful to my husband and love of my life, Amirali, for being a constant source of support and encouragement during the challenges of graduate school and life. I would also like to thank my family, especially my parents and brother, for their unconditional love and support. Their faith in me has been the key asset in achieving my goals.

Finally, I'm grateful for all the financial support I received. My research was supported by the National Science Foundation CDS&E Grant no. OAC-1808591. It was also sponsored by the U.S. Army Combat Capabilities Development Command Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-13-2-0045 (ARL Cyber Security CRA). Also, GPUs used for this research were donated by NVIDIA Corporation.

To my beloved husband and my lovely parents for supporting and encouraging me every step of the way.

ABSTRACT OF THE DISSERTATION

Defending Against Adversarial Attacks With Low-Rank Factorization

by

Negin Entezari

Doctor of Philosophy, Graduate Program in Computer Science
University of California, Riverside, March 2022
Dr. Evangelos Papalexakis, Chairperson

Despite the popularity and success of deep learning architectures in recent years, they have shown to be vulnerable to adversarial attacks. Utilizing machine learning methods that are vulnerable to adversarial attacks has raised many concerns, especially in safety-critical domains. Self-driving vehicles and medical imaging are examples of systems where perturbations in the data can have dramatic and irreparable impacts as the malfunction of the system may result in death or serious injuries. Fake product reviews are another example of deliberate perturbations that negatively impact everyone's life by fooling people into purchasing fake and potentially dangerous products. Therefore, understanding these vulnerabilities and designing robust defense mechanisms against perturbations could benefit a large group of society.

Recent studies have addressed these concerns and conducted research to analyze the vulnerability of machine learning algorithms and develop defense techniques and methods that are more robust to attacks. However, the general properties of adversarial attacks have not been investigated. In this dissertation, we analyze the properties of adversarial attacks in different domains, including graphs, images, and recommender systems. Our goal is to identify a unifying theme to propose a general defense mechanism applicable to various domains. Attackers try to achieve their malicious intentions by crafting perturbations into data while attempting to remain unnoticeable. In the image domain, attackers add noise to the high-frequency spectrum of the images that are not perceptible by the human eyes. Unlike images, in other domains like graphs and recommender systems, changes cannot be easily noticed by looking at the data. However, attackers try to preserve some key features and measurable statistics of the original data to remain unnoticeable. This could be achieved by preserving the degree distribution of the nodes in graphs and rating distributions in recommender systems. These imperceptibility constraints prohibit attackers from making significant changes to the data. Therefore, the footprint of the attack is subtle compared to the footprint of the structure of the entire data. In other words, the impact of the adversaries is noticeable mainly in the high-frequency spectrum of the data.

In this dissertation, we identify a unifying theme in adversarial attacks across different applications domains (graph node classification, image classification, and recommender systems). More specifically, we observe that in all three domains, the attack manifests in high ranks of the matrix or tensor representing the data. Motivated by this unifying theme, we propose low-rank solutions in different domains to alleviate the negative impact of adversaries. Finally, as a case study, we propose a low-rank tensor-based method to improve the quality of complementary basket recommendations, where the goal is to recommend products that are frequently purchased with the items in the users' shopping cart.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

In recent years Deep Neural Networks (DNNs) have gained tremendous success in a variety of tasks, including computer vision, speech recognition, graph analysis, and many more. Despite the popularity and success of deep learning architectures in recent years, they have shown to be vulnerable to adversarial attacks [255]. Recent studies have shown that Machine learning models often suffer from vulnerabilities to adversarial perturbations [76]. Subtle perturbations of the data can be imperceptible yet lead to wrong results. Even when the attacker does not have full knowledge of the network architecture, they are still able to perturb the data and affect the learning outcome.

Utilizing machine learning methods which are vulnerable to adversarial attacks have raised many concerns. Recent studies have addressed this concern and conducted research to analyze vulnerability of machine learning algorithms and also develop defense techniques and methods that are more robust to attacks [233]. However, the general properties of adversarial attacks has not been investigated. Attackers attempt to craft perturbations such that they are imperceptible, however, they can significantly deteriorate the performance of machine learning models. Our goal is to understand common properties of adversarial attacks in different domains including graphs, images, and recommender systems and propose a general defense mechanism applicable to various domains.

The major contribution of this dissertation is the establishment of a link between a dataset that contains adversarially attacked data points and the *rank* of that data. More specifically, we identify a recurring theme of adversarial attacks manifesting in high-rank components of the data. This pattern persists across different applications: graphs, images, and recommender systems. Informed by this pattern, we develop low-rank-based defenses for those applications. The methodological contributions of this dissertation are:

- Defense against adversarial attacks on graphs
- Defense against adversarial attacks on images
- Defense against adversarial attacks on recommender systems
- Low-rank tensor-based complementary product recommendation

In Chapter 2, we attempt to address the problem of adversarial attack for node classification task. Our proposed methods utilize low-rank factorization of input data to discard attacks. For graph data, we show that the adversaries affect higher ranks in the singular value spectrum of the graph and therefore a low-rank approximation of the graph can significantly alleviate the attacks. on the CiteSeer dataset, our proposed defense mechanism is able to reduce the success rate of the attack from 98% to 36%.

In Chapter 3, we tackle the problem of adversarial attack for image classification task. For image data, perturbations are mostly crafted in high-frequency domain to remain unnoticeable to human eye, therefore by performing tensor decomposition techniques and low-rank reconstruction of images we could improve the performance of deep convolutional network for the task of image classification. Our tensor-based defense mechanism outperforms the state-of-the-art method named SLQ by 14% against Fast Gradient Descent (FGSM) adversarial attacks, while maintaining comparable speed.

In Part III, we evaluate the impact of low-rank solution on the performance of recommender systems from two different aspects. In Chapter 4, we consider low-rank solutions from adversarial point of view. In recommender systems domain, we observe that low-rank reconstructions and/or transformation of the attacked data has a significant alleviating effect on the attack. We also demonstrate that a simple classifier is able to learn to detect fake users from real users and can successfully discard them from the dataset. This observation elaborates the fact that the threat model does not generate fake users that mimic the same behavior of real users and can be easily distinguished from real users' behavior. We also examine how transforming latent factors of the matrix factorization model into a low-dimensional space impacts its performance. Furthermore, we combine fake users from both attacks to examine how our proposed defense is able to defend against multiple attacks at the same time. Local low-rank reconstruction was able to reduce the hit ratio of target items from 23.54% to 15.69% while the overall performance of the recommender system was preserved. In Chapter 5, we we utilize low-rank tensor-based solution to solve the problem of complementary product recommendation. To satisfy the customers' needs, it is vital to provide relevant personalized recommendations and ease the customers' shopping experience. In Chapter 5, we propose a tensor-based method that utilizes a three-mode

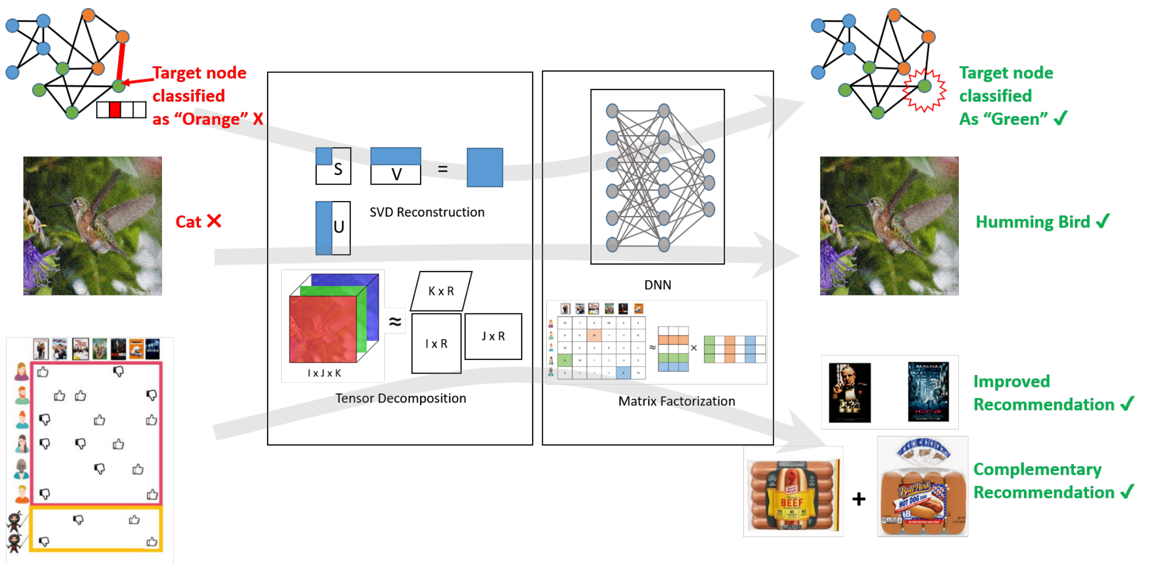


Figure 1.1: System Overview: low-rank solution to defend against adversarial attacks on graphs, images, and recommender systems that leads to improvements in tasks including node classification, image classification, product recommendation, and complementary product recommendation.

tensor to represent product-to-product relations for users and applies tensor decomposition techniques to jointly learn user and product embeddings that can be used to infer within-basket recommendations. Products co-purchased in a single transaction are modeled in the form of a tensor. Then, we leverage RESCAL tensor decomposition technique to capture the latent factors that reveal the inherent user and product interactions. On the Instacart dataset, our proposed tensor-based method achieves a recall@10 of 0.192, whereas recall@10 for triple2vec, which is the state-of-the-art, is 0.149. Figure 1.1 illustrates an overview of our proposed methods in this dissertation.

Part I

# Node Classification

Chapter 2

# Defense Against Adversarial Attacks on Graphs

Chapter based on material published in WSDM 2020 [93].

## 2.1 Introduction

Graphs are widely used because of their strength in representing real-world data in many domains, such as social networks, biological networks, and citation networks. Due to the ubiquity of graphs, analyzing them has gained significant attention in recent years. An important task in analyzing graph data is node classification. Given a partially labeled (attributed) graph, the goal is to classify the entire graph and predict the labels of the unknown nodes [48]. Graph representation learning [122, 225] and deep learning techniques [155, 228] have shown outstanding results in addressing the problem of node classification.

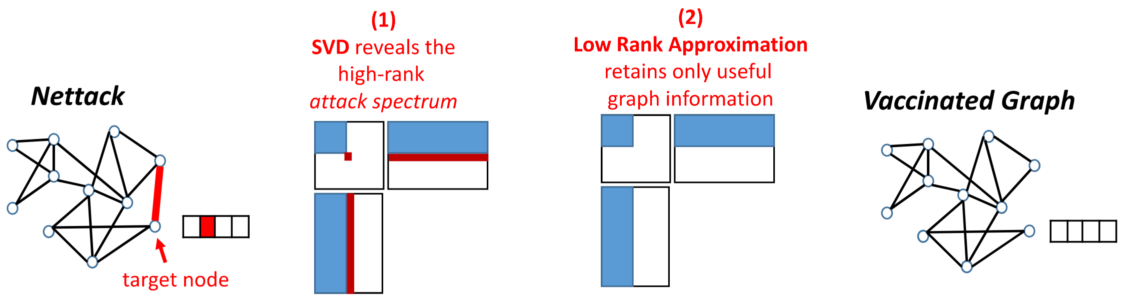


Figure 2.1: A quick sketch of our proposed vaccination: Taking the SVD of the graph reveals the spectrum of the attack and the healthy parts of the graph. Based on our extensive empirical observations on the high-rankness of NETTACK, we retain a truncated SVD that contains only the top-$k$ singular values for the graph, and reconstruct the graph from them. The output is the vaccinated graph.

Deep Neural Networks deployed for analyzing graphs are also affected by adversarial attacks. There are only a few studies investigating adversarial attacks on graph data [288, 75, 51, 252]. Graph Convolutional Networks (GCN) have shown great success in node classification task because of their non-linear nature and exploiting relational information of nodes [155]. Despite their success, they suffer from vulnerabilities against small perturbations. Changes to one node can lead to misclassification of other nodes in the graph [288]. Writing wrong or biased reviews on websites like Amazon and fake users on social networks are examples of adversarial attacks on graph data. Such activities aim to mislead machine learning techniques. Therefore, adversarial machine learning studies play a crucial role in graph domain and their goal is to detect and defend attacks and also introduce techniques that are more robust against perturbations.

One of the most prominent studies on generating adversarial attacks for graphs is called NETTACK [288] proposed by Zügner et al. Their work shows that the classification performance of GCN drops significantly when NETTACK perturbations target a node. In this dissertation, I investigate the properties of poisoning adversarial attacks generated by the NETTACK algorithm and propose a method to defend against attacks and "vaccinate" the network. The term "vaccinate" was first introduced in [77] to describe a network equipped with defense mechanism against adversarial attacks.

## 2.2 Related Work

### 2.2.1 Graph Representation Learning Methods

Our work focuses on defending adversarial attacks on graphs and we evaluate the robustness of a node embeddings method against the attacks. In this section, we briefly explain node embeddings for the task of node classification.

Recent years have witnessed an explosion in studying the problem of network representation learning. This interest is stimulated by the "relatively" new advancements in natural language processing (NLP) domain [197, 171, 196]. Specifically, the SkipGram model [196] that has been largely adopted in developing network representation learning techniques. DeepWalk [225], node2vec [122], and Walklets perozzi2016walklets are amongst the methods that employ the SkipGram model for node representation learning after identifying node neighborhoods using the intuition of random walks and they have shown to be very successful for the task of node classification.

A recent study has proposed a tensor-based node embedding method that utilizes tensor decomposition to learn network latent features using the CP decomposition of tensors [30].

### 2.2.2 Adversarial Attacks for Graph Data

Research on adversarial attacks in machine learning has received lots of attention in recent years [50, 192]. Adversarial attacks deliberately attempt to attenuate the performance of machine learning algorithms by performing small and unnoticeable changes to the input. Most of the researches on adversarial machine learning are focused on algorithms to fool deep neural networks, mainly for the task of image classification [255, 169, 200].

Recently, a few work have investigated adversarial attacks on graph data [288, 51, 252, 75]. Zügner et al. [288] perform structure and feature perturbations on attributed graphs by an algorithm called Nettack. They generate unnoticeable perturbations by preserving graph's degree distribution and features co-occurrences. The performance of Nettack on attacking GCN shows that it can successfully fool GCN and lead to misclassification of the target node.

In another study, Dai et al. [75] proposed a reinforcement learning based attack that generates structure perturbations with full or limited information about the target classifier. Their approach has shown to be successful for supervised node classification problem. They also claim that adding adversarial examples during training can help to defend the attacks.

The other group of studies, investigate the effects of adversarial attacks on unsupervised node embeddings [288, 51, 252]. In [288], they transferred their attack model to

DeepWalk embeddings [225] and observed that the performance of DeepWalk drops on a perturbed graph.

Knowing the fact that graph neural networks and node embeddings are highly vulnerable to adversarial attack, there is an urgent need to design defense mechanism or effective methods that are more robust against attacks. There are some studies on defense techniques in tasks like image classification [77, 49]. In [77], JPEG compression has been used to “vaccinate” deep neural network. The idea is that adversarial attacks on images add noise to high-frequency spectrum so that the noise is visually imperceptible, therefore, JPEG compression can greatly destroy them. In another study, Bhagoji et al. [49], proposed a defense mechanism that utilizes Principal Component Analysis (PCA) for dimensionality reduction.

When it comes to physical adversaries and the goal is the detection of dense block-like behavior, which points to fraud (e.g., fraudulent Twitter followers), [237] leverages Singular Value Decomposition (SVD) and low-rank approximation of adjacency matrix of users in Twitter and Amazon to detect suspicious behavior. They analyze the impacts of evasion attacks in their study, where the malicious data are modified at test time to bypass the result. Another group of attacks are poisoning attacks which perturb the training instances and the classification model is retrained on the perturbed data. In our study, we explore poisoning attacks on graph data and we propose a mechanism to defend the attacks.

## 2.3 Preliminaries

In this section, we describe concepts and notations used in the chapter.

### 2.3.1 Singular Value Decomposition

Singular Value Decomposition (SVD) is one of the most popular matrix decomposition techniques. SVD is a widely used tool to decompose a matrix into sum of rank-1 matrices. Let $A \in \mathbb{R}^{I \times J}$ be a real-valued matrix. The SVD of $A$ is computed as follows:

$$A = U \Sigma V^T \tag{2.1}$$

where $U \in \mathbb{R}^{I \times I}$ and $V \in \mathbb{R}^{J \times J}$ are orthogonal matrices. Column of $U$ are called the left singular vectors and columns of $V$ are the right singular vector. $\Sigma \in \mathbb{R}^{I \times J}$ is a non-negative diagonal matrix such that $\Sigma_{i,i} = \sigma_i$ where $\sigma_i$ is the $i$th singular value and $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_{min(I,J)}$.

The SVD is an elegant tool to compute the best rank-$r$ approximation of matrix $A$. The rank-$r$ approximation of $A$ is computed as follows:

$$A_r = U_r \Sigma_r V_r^T = \sum_{i=1}^{r} u_i \sigma_i v_i^T \tag{2.2}$$

where $A_r$ is the rank-$r$ approximation of $A$ derived from SVD of $A$. $U_r$ and $V_r$ are the matrices containing the top $r$ singular vectors and $\Sigma_r$ is the diagonal matrix containing only the $r$ singular values.

According to Eckart-Young-Mirsky theorem [91], $A_r$ is the optimal rank-$r$ approximation of matrix $A$. For any rank-$r$ matrix $B$, the following holds:

$$\|A - A_r\|_F \leq \|A - B\|_F \tag{2.3}$$

### 2.3.2 Tensors

A tensor, denoted by $\underline{\mathbf{X}}$, is a multidimensional matrix. The order of a tensor is the number of modes/ways which is the number of indices required to index the tensor [217]. In this chapter, we deal with three-mode tensors. Given a three-mode tensor $X \in \mathbb{R}^{I \times J \times K}$ its CP decomposition (also know as CANDECOMP/PARAFAC) [61, 129] is defined as a sum of rank-1 tensors and is formulated as follows:

$$\underline{\mathbf{X}} \approx \sum_{r=1}^{R} a_r \circ b_r \circ c_r \tag{2.4}$$

where $a_r \in \mathbb{R}^I, b_r \in \mathbb{R}^J, c_r \in \mathbb{R}^K$, and their three-way outer product is computed as $(a_r \circ b_r \circ c_r)(i,j,k) = a_r(i)b_r(j)c_r(k)$. The minimal value of $R$ is called the tensor *rank*. For a more compact representation, CP decomposition is usually represented by the *factor matrices* $A \in \mathbb{R}^{I \times R}$, $B \in \mathbb{R}^{J \times R}$, and $C \in \mathbb{R}^{K \times R}$ where $a_r$, $b_r$ and $c_r$ are the $r$th columns of $A$,$B$, and $C$ respectively.

$$\underline{\mathbf{X}} = A \circ B \circ C \tag{2.5}$$

For more details about tensors and tensor decomposition, we refer the interested reader to [217, 160].

## 2.4 Proposed Method

In this section, we present our low-rank matrix approximation method to defend adversarial attacks for graph data. In Section 2.4.1, we briefly explain the method to generate adversarial attacks for graph data, known as Nettack [288] and we examine the characteristics of the attacks generated by this approach. We show that these attacks impose high-rank changes to the graph which can be greatly ignored by discarding the high-rank components of the graph. In Section 2.4.2, we present two low-rank methods that are robust to adversarial attacks generated by NETTACK. Moreover, in Section 2.4.3 we propose a low-rank attack and investigate its characteristics compared to NETTACK.

### 2.4.1 NETTACK: a High-Rank Attack

Recently, Zügner et al. [288] introduced an algorithm to generate adversarial attacks for attributed graphs to fool Graph Convolutional Networks. Given an attributed graph $G = (A, X)$, where $A \in \{0,1\}^{N \times N}$ is the undirected adjacency matrix and $X \in \{0,1\}^{N \times D}$ represents the nodes' feature matrix, the goal is to perform small perturbations on the graph $G^{(0)} = (A^{(0)}, X^{(0)})$, so that the result graph $G' = (A', X')$ has lower classification performance and leads to misclassification of the target node. Structure perturbations refer to the changes to the adjacency matrix $A$, while feature perturbations refer to the changes to the feature matrix $X$ . NETTACK produces unnoticeable perturbations by imposing some restrictions to ensure that the attack preserves graph structure and node features. To generate unnoticeable structure perturbations, attacks that preserve the degree distribution of the graph are considered unnoticeable. Whereas, to generate unnoticeable feature perturbations, co-occurrence of the features is taken into consideration. We refer the interested reader to [288] for more details.

**<u>Intuition</u>:** Nettack perturbations affect small number of nodes. Thus, the footprint of the spectrum of this attack will be comparably smaller than the footprint of the regular structure in the graph, therefore, the attack will likely appear in small singular values, corresponding to higher ranks in the singular value spectrum.

To experimentally validate our intuition and understand the characteristics of the perturbations generated by the Nettack model, we examined the adjacency and feature matrices before and after the attack. We plotted the singular values of matrices for the clean and perturbed graphs to visualize the differences. Figure 2.2 illustrates the singular values of the adjacency matrix on semi-logarithmic scale. The singular values shown in Figure 2.2 correspond to singular values of the adjacency matrix before and after one single attack on the target node. Singular values of the clean and attacked matrices are mainly different at higher ranks. We visualized singular values of adjacency and feature matrices for multiple

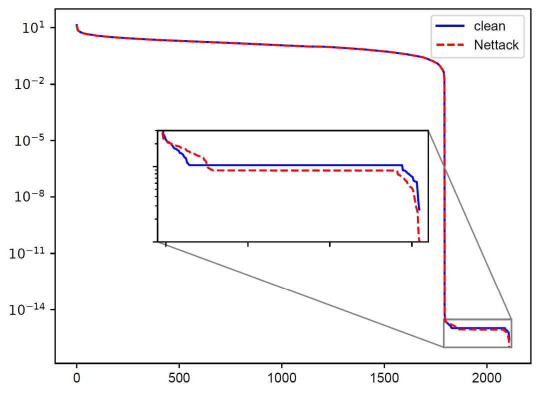


Figure 2.2: Singular values of adjacency matrix before and after the attack in a semi-logarithmic scale

attacks and observed that singular values are very close at lower ranks but vary at higher ranks.

In Section 2.4.2, we take advantage of this intuition and present two low-rank solutions that can effectively resist against NETTACK.

### 2.4.2 Low-Rank Solutions to Resist Attacks

#### Vaccinating GCN with Low-Rank Approximation

To discard the high-rank perturbations generated by NETTACK, we compute the low-rank approximation of the adjacency and feature matrices derived from their SVD decomposition according to Equation 2.2. We then retrain GCN with the low-rank approximation matrices. With a proper choice of $r$, the rank-$r$ approximation of the attacked graph can boost the performance of GCN and achieve a performance close to the performance of GCN on the clean graph.

Let $A$ and $A'$ be the adjacency matrices of the clean and attacked graphs respectively. $\delta A = A' - A$ is the difference between the clean and attacked graphs after a series of perturbations. These are the edges added to the clean graph or removed from it as a result of the attack. We compute the SVD of $A$ and $\delta A$ as follows:

$$A = U \Sigma V^T \tag{2.6}$$

$$\delta A = U_\delta \Sigma_\delta V_\delta^T \tag{2.7}$$

Let $n$ be the number of perturbations performed during one attack on the target node $v_0$. According to [288], $n = d_{v_0} + 2$, where $d_{v_0}$ is the degree of the target node. Leveraging the proof from [237], the leading singular value of $\delta A$ is computed as follows:

$$\sigma_{\delta 1} = \sqrt{n} = \sqrt{d_{v_0} + 2} \tag{2.8}$$

In a rank-$r$ approximation of the attacked graph, singular values smaller than $\sigma_r$ are discarded. Therefore, if $\sigma_{\delta 1}$ is smaller than $\sigma_r$, the perturbations will get eliminated. The goal is to pick rank $r$ so that with a high probability the following holds:

$$\begin{aligned} &\sigma_r > \sigma_{\delta 1} \\ &\sigma_r > \sqrt{d_{v_0} + 2} \end{aligned} \tag{2.9}$$

$$d_{v_0} < \sigma_r^2 - 2 \tag{2.10}$$

In other words, a rank-$r$ approximation may not detect attacks on target nodes with degree greater than $\sigma_r^2 - 2$. Formally, we pose the following problem:

**Problem:** Given an input graph adjacency matrix $A$ with $r$th largest singular value $\sigma_r$, find the value of $r$ so that the probability of the nodes with degree greater than $\sigma_r^2$ is less than a given threshold $\tau$:

$$Pr(X \geq \sigma_r^2) < \tau \tag{2.11}$$

Degree distribution of graphs in real networks has a power-law form. We can write the degree distribution of a graph in form of a discrete power-law with parameter $\alpha$:

$$\begin{aligned} p(d) = Pr(X = d) &= \frac{d^{-\alpha}}{\sum\limits_{k=d_{min}}^{d_{max}} k^{-\alpha}} \\ &= \frac{d^{-\alpha}}{\sum\limits_{k=0}^{d_{max}-d_{min}} (k + d_{min})^{-\alpha}} \\ &= \frac{d^{-\alpha}}{\zeta(\alpha, d_{min}) - \zeta(\alpha, d_{max} + 1)} \end{aligned} \tag{2.12}$$

where $\zeta(\alpha, x) = \sum_{k=0}^{\infty}(k + x)^{-\alpha}$ is the Hurwitz zeta function. $d_{min}$ is the minimum degree of a node required to be considered in the power-law distribution and $d_{max}$ is the maximum degree in the graph. Therefore, for a given graph $G$, we can write its degree distribution as follows:

$$p(d)_{d\in\mathcal{D}_G} \approx \frac{d^{-\alpha}}{\zeta(\alpha, d_{min}) - \zeta(\alpha, d_{max} + 1)} \tag{2.13}$$

where $\mathcal{D}_G = \{d_v^G | v \in \mathcal{V}, d_v^G \geq d_{min}\}$ is the list of node degrees in the graph $G$. Clauset et al. [72] drived an approximate expression to estimate the scaling parameter $\alpha$ for a discrete power-law distribution. For graph $G$:

$$\alpha \approx 1 + |\mathcal{D}_G|. \left[ \sum_{d_i\in\mathcal{D}_G} log \frac{d_i}{d_{min} - \frac{1}{2}} \right]^{-1} \tag{2.14}$$

To find the solution for Equation 2.11, we compute the reverse cumulative probability of power-law as follows:

$$\begin{aligned} Pr(X \geq d) &= \frac{\sum\limits_{d_i=d}^{d_{max}} Pr(X = d_i)}{\zeta(\alpha, d_{min}) - \zeta(\alpha, d_{max} + 1)} \\ &= \frac{\sum\limits_{d_i=d}^{d_{max}} d_i^{-\alpha}}{\zeta(\alpha, d_{min}) - \zeta(\alpha, d_{max} + 1)} \\ &= \frac{\zeta(\alpha, d) - \zeta(\alpha, d_{max} + 1)}{\zeta(\alpha, d_{min}) - \zeta(\alpha, d_{max} + 1)} \end{aligned} \tag{2.15}$$

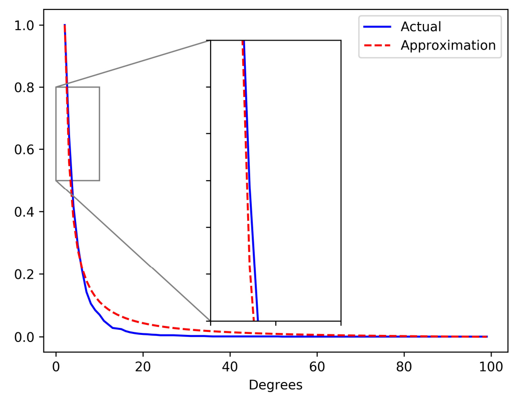


Figure 2.3: Reverse cumulative degree distribution of CiteSeer dataset

And for graph $G$:

$$Pr(X \geq d)_{d \in \mathcal{D}_G} \approx \frac{\zeta(\alpha, d) - \zeta(\alpha, d_{max} + 1)}{\zeta(\alpha, d_{min}) - \zeta(\alpha, d_{max} + 1)} \tag{2.16}$$

From equations 2.11 and 2.16, we can formulate the problem as follows:

$$Pr(X \geq \sigma_r^2)_{\mathcal{D}_G} \approx \frac{\zeta(\alpha, \sigma_r^2) - \zeta(\alpha, d_{max} + 1)}{\zeta(\alpha, d_{min}) - \zeta(\alpha, d_{max} + 1)} < \tau \tag{2.17}$$

Figure 2.3 shows the actual reverse cumulative degree distribution of the CiteSeer dataset vs. its approximation using Equation 2.16. The behavior on Cora-ML and PoliticalBlogs datasets is very similar to Figure 2.3 as well. As illustrated in Figure 2.3, the reverse cumulative probability quickly drops to zero. On CiteSeer dataset, the probability of degrees greater than 20 is less than 1%. Thus, a low-rank approximation of the graph with high probability can eliminate the perturbations.

In the experimental evaluations that follows, we evaluate Equation 2.11 for different values of rank $r$ on the real-world datasets and experimentally show that how big $r$ needs to be to successfully vaccinate GCN against adversarial perturbations.

## Robust Tensor-Based Node Embeddings: t-PINE

Recently, Al-Sayouri et al. [30] proposed a tensor-based node embedding method that utilizes tensor's CP decomposition to capture the relations between nodes using low-dimensional latent components. They examined the performance of t-PINE in the context of node embeddings, however the robustness of t-PINE has not been evaluated in an adversarial context. Due to the inherent low-rank nature of t-PINE, it is a good candidate to defend high-rank perturbations generated by NETTACK. Here, we briefly explain t-PINE:

t-PINE jointly encodes explicit[1] and implicit[2] network structure [30] using CP decomposition [61], which greatly allows for a systematic exploration of higher-order proximities. Due to the use of multi-aspect data, t-PINE forms a three-mode tensor to represent a network which has two slices: (1) The adjacency matrix, and (2) $K$-nearest neighbor matrix computed for the feature matrix. Then, tensor $\underline{\mathbf{x}} \in \mathbb{R}^{N \times N \times 2}$ is decomposed using CP decomposition as given in Equation 2.4. The CP model is solved using the Alternating Least Squares (ALS) algorithm [61]. For a predefined parameter $d$ which is the embedding dimension, tensor $\underline{\mathbf{X}}$ is decomposed as:

$$\mathcal{L} \approx \min \|\underline{\mathbf{X}} - A \circ B \circ C\|_F^2 \tag{2.18}$$

where $A \in \mathbb{R}^{N \times d}$, $B \in \mathbb{R}^{N \times d}$, and $C \in \mathbb{R}^{2 \times d}$ are the factor matrices. When CP decomposition is used in multi-label classification problem, the *tensor rank* $R$ denotes the number of classes, however, in t-PINE, $R = d$, indicates the embedding dimensionality, as the CP decomposition is tailored for representation learning purpose.

[1]Refers to network first-order proximity connections

[2]Refers to second- or higher-order proximity connections

In contrast to state-of-the-art approaches [225, 122, 226, 274, 126], t-PINE yields highly predictable representations on different multi-label classification problems. Further, it generates nicely interpretable embeddings, where we can understand how each view contributes to the learned representation vectors. For more details, we refer the reader to [30].

### 2.4.3 *LowBlow*: a Low-Rank Attack

As explained in Section 2.4.1, Nettack perturbations cause changes to high-rank singular values which could be defended using a low-rank approximation of the graph. Now a question that might arise is that what happens if we have a low-rank attack. Is a low-rank perturbation able to successfully attack the graph and fool GCN and if so, are we able to defend it using our proposed low-rank mechanism? Is the low-rank attack still unnoticeable? We will answer these questions in the experimental evaluations. In this section, we will manipulate the Nettack perturbations so that it results to low-rank attacks that can hinder the performance of both GCN and t-PINE.

To generate a low-rank attack, we replace some of the most significant singular values and vectors of $\delta A$ with the corresponding singular values and vectors of $A$. In other words, singular values of $\delta A$ within range $[i, j]$ $(\sigma_{\delta i}, ..., \sigma_{\delta j})$ are replaced by singular values of $A$ $(\sigma_i, ..., \sigma_j)$. $i$ and $j$ are relatively small values so that the corresponding singular values are significant (e.g. less than 100). Then we reconstruct the low-rank $\delta A$ from altered singular values and vectors. Adding this low-rank $\delta A$ to the clean adjacency matrix $A$ will result to a low-rank attack that is able to perturb GCN and t-PINE.

$$A'_{low-rank} = A + \delta A_{low-rank} \tag{2.19}$$

We compute the low-rank perturbations on the feature matrix in an analogous way. Let $F$ and $F'$ be the feature matrices of the clean and attacked graphs, respectively. $\delta F = F' - F$ is the matrix representing features added to/removed from the original feature matrix $F$. After computing the SVD of $F$ and $\delta F$, we replace some of the most significant singular values and vectors of $\delta F$ with the corresponding singular values and vectors of $F$. $\delta F_{low-rank}$ is reconstructed using these modified singular values and vectors. To get the low-rank feature perturbations, we add $\delta F_{low-rank}$ to $F$.

$$F'_{low-rank} = F + \delta F_{low-rank} \tag{2.20}$$

In the experimental evaluation that follows, we verify the effectiveness of *LowBlow*, and we also evaluate the extent to which *LowBlow* alters the perception of the graph to an observer, in the form of the node degree distributions.

## 2.5 Experiments

### 2.5.1 Datasets and Experiment Setup

**Datasets:** In order to compare our results to the adversarial attack paper [288], we use the same datasets in our experiments. The datasets are CiteSeer [116], Cora-ML[52], and PoliticalBlogs [23]. Cora-ML is the subset of machine learning papers from the well-

| **Dataset** | **$\lvert V \rvert$** | **$\lvert E \rvert$** | **$\lvert V_{LCC} \rvert$** | **$\lvert E_{LCC} \rvert$** | **Classes** |
|---|---|---|---|---|---|
| CiteSeer [116] | 3312 | 4715 | 2110 | 3757 | 6 |
| Cora-ML[52] | 2995 | 8416 | 2810 | 7981 | 7 |
| PoliticalBlogs[23] | 1490 | 19025 | 1222 | 16714 | 2 |

Table 2.1: Datasets descriptions

know Cora dataset [191]. Table 2.1 provides the statistics for each dataset. All experiments are performed on the largest connected component (LCC) of the graphs.

**Setup:** In section 2.4.1, we showed that Nettack perturbations are of high-rank. To further investigate our intuition, we follow the same procedure as described in the Nettack paper [288]. We split the network in labeled (20%) and unlabeled nodes(80%). half of the labeled data is used for training and the other half is used for validation in the process of training the GCN model. We perform five iterations where at each iteration a different random splits of data is generated. We first train the GCN surrogate model on the labeled data and then we select 40 target nodes from test set with the following conditions:

- 20 nodes which are correctly classified: 10 of them have the highest classification margins and 10 of them have the lowest margin.
- 20 random nodes

According to the algorithm proposed in [288], there are two different ways to attack a target node : direct attack called Nettack, and influence attack called Nettack-In which attacks a node indirectly. In our experiments we only consider attacking each target node directly, as it is a stronger attack compared to an indirect attack. We also combine structure and feature perturbations which leads to a greater performance loss. To evaluate the effectiveness of the attack, we compute $X = Z^*_{v0,c_{old}} - max_{c \neq c_{old}} Z^*_{v0,c}$ where $Z$ is the class probabilities and $c_{old}$ is the ground truth label of the target node. $X$ is called the classification margin. A successful attack leads to lower values of $X$ and a negative value means the target node has been successfully misclassified.

### 2.5.2 Vaccinating GCN with Low-Rank Approximation

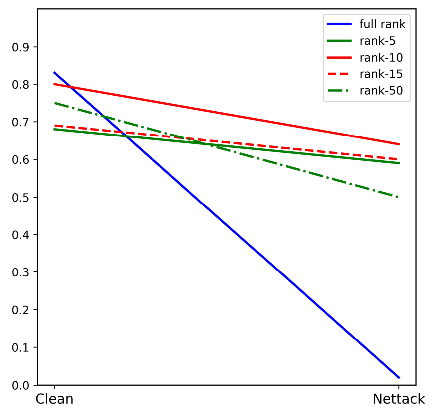
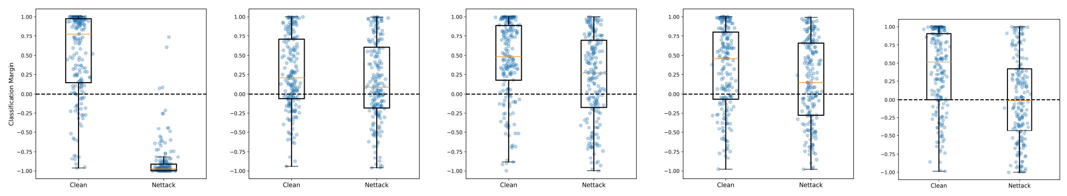


Figure 2.4: Fraction of target nodes correctly classified after vaccinating GCN on CiteSeer

In Section 2.4.2, we presented the low-rank approximation to defend Nettack perturbations. In this section, we analyze the performance of our defense mechanism. To this end, we examine different values of rank $r = 5, 10, 15$, and 50 to compute the approx-



(a) Unvaccinated (b) rank-5 (c) rank-10 (d) rank-15 (e) rank-50

Figure 2.5: Vaccinating GCN against Nettack

| **Method** | **CiteSeer** | **Cora-ML** | **PoliticalBlogs** |
|---|---|---|---|
| GCN - Clean | 0.83 | 0.82 | 0.90 |
| GCN - NETTACK | 0.02 | 0.01 | 0.09 |
| Vaccinated - Clean | 0.80 | 0.76 | 0.84 |
| Vaccinated - NETTACK | 0.64 | 0.59 | 0.62 |

Table 2.2: Vaccinating GCN against NETTACK. Fraction of target nodes that are correctly classified is reported.

imations. Figure 2.4 shows that the fraction of target nodes correctly classified after the attack drops significantly with the full-rank attacked matrices. However, using the low-rank SVD approximation, this number is close to the fraction of correctly classified nodes on the clean graph. Figures 2.4 and 2.5 illustrate that using rank-10 approximation of the adjacency and feature matrices we are able to significantly alleviate the effects of NETTACK. Only 10 singular values/vectors is sufficient to have a robust approximation of the graph structure and features and vaccinate GCN against attacks. As we discussed in Section 2.4.2 and Equation 2.11, if $Pr(X \geq \sigma_{10}^2)$ is less than a threshold $\tau$, with a high probability we discard perturbations. $Pr(X \geq \sigma_{10}^2)$ is 0.0013, 0.0019, and 0.0037 on CiteSeer, Cora-ML, and PoliticalBlogs, respectively. These probabilities are nearly zero which shows that a rank-10 approximation of the graph ignores perturbations with a very high probability.

We performed this experiment on all three datasets and observed that $r = 10$ produces the best results. Table 2.2 shows the results. Here, for brevity, we only report the results for $r = 10$.

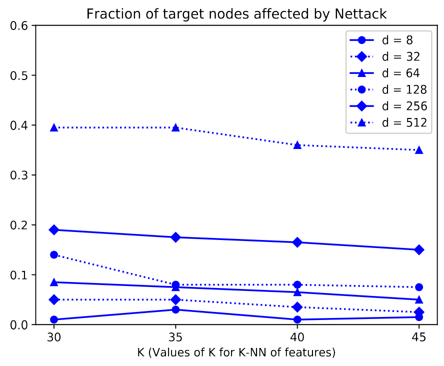
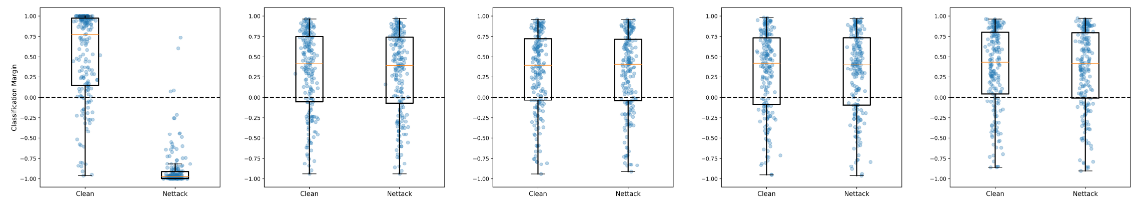


Figure 2.6: Robustness of t-PINE against NETTACK for different embedding dimensions and K on CiteSeer

### 2.5.3 Transferring Adversarial Attacks to t-PINE Embeddings

To evaluate the transferability of adversarial attacks to the tensor-based embeddings, we pursue the experiment as explained in Section 2.5.1 . After every attack, we compute the t-PINE embeddings where the tensor slices are the attacked adjacency and feature matrices.

We perform the experiment for different values of embedding dimensions $d$ and $K$ to examine their effects on the robustness of t-PINE. Figure 2.6 shows the fraction of target nodes that their prediction changed after the attack. This does not necessarily mean that these nodes were correctly classified before the attack, but it shows that the NETTACK perturbations were able to change the prediction of the target nodes from one class to another. NETTACK might have even changed a node's prediction from a wrong class to the correct one. The plot shows that lower dimensions of t-PINE are more robust against NETTACK. As the embedding dimension gets larger, more nodes are affected by the attacks. t-PINE performs very robust against the attacks. Even at dimension 512, less than 40%



(a) GCN (b) $K = 30$ (c) $K = 35$ (d) $K = 40$ (e) $K = 45$

Figure 2.7: Poisoning of t-PINE with Nettack on CiteSeer. The embedding dimension is 32.

of target nodes are affected by the attacks. At lower dimensions, CP-decomposition can greatly discard affected components of graph by the attack.

On the other hand, choice of $K$ does not have a significant effect on the robustness of t-PINE. We evaluated different values of $K$ and observed that at a fixed dimension, performance slightly improves for bigger values of $K$. However, the larger $K$ is, t-PINE's runtime increases. The improvement over larger values of $K$ is negligible. Therefore, for the rest of the experiments, we only report the t-PINE results for $d = 32$ and $K = 30$ that leads to robust results in a better runtime.

Figure 2.7 shows the result of transferring Nettack perturbations to t-PINE for $d = 32$ and $K = 30$. The plot shows that t-PINE is very robust to the attacks and the classification margins before and after the attack has remained nearly unchanged.

| | **Method** | **CiteSeer** | **Cora-ML** | **PoliticalBlogs** |
|---|---|---|---|---|
| GCN | Clean | 0.83 | 0.82 | 0.90 |
| | Nettack | 0.02 | 0.01 | 0.06 |
| t-PINE | Clean | 0.74 | 0.68 | 0.87 |
| | Nettack | 0.72 | 0.64 | 0.30 |

Table 2.3: Transferring Nettack to t-PINE embeddings. For t-PINE, fraction of target nodes correctly classified after the attack is very close to values on the clean graph.

In Table 2.3, we summarize the results of transferring Nettack perturbation to t-PINE for different datasets. The values reported in the table are the fraction of target nodes that get correctly classified. For t-PINE the values on clean and perturbed graphs are very close for CiteSeer and Cora-ML datasets. However, on PoliticalBlogs, the performance of t-PINE has dropped with Nettack perturbations. The degree of target nodes in PoliticalBlogs dataset are relatively larger compared to the other datasets. In our experiments, we set the number of perturbations to a target node relevant to its degree. Therefore, in the PoliticalBlogs dataset, we perform a larger number of perturbation and this could be the reason to why the performance of t-PINE drops when facing Nettack perturbations.

### 2.5.4 *LowBlow*: A Low-Rank Attack

Here, we investigate the influence of the proposed low-rank attack, *LowBlow* on GCN and t-PINE. To evaluate the effects of *LowBlow*, we compute the perturbed adjacency and feature matrices as in Equations 2.19 and 2.20. Then we retrain GCN model with the perturbed matrices. We also compute the t-PINE embeddings for the perturbed matrices. *LowBlow* significantly decreases the performance of GCN. It is also able to attack t-PINE, however, it is less successful compared to perturbing GCN.

In addition, we examined our defense mechanism against *LowBlow*. We used a rank-10 approximation of graph to vaccinate it. In Table 2.4, we summarize the results for all datasets. Vaccinating GCN has improved its performance but it decreased the performance of t-PINE on CiteSeer and Cora. We observed that for a smaller embedding dimension e.g

|  | Method | CiteSeer | Cora-ML | PoliticalBlogs |
|---|---|---|---|---|
| GCN | Clean | 0.83 | 0.82 | 0.90 |
| GCN | NETTACK | 0.02 | 0.01 | 0.06 |
| GCN | *LowBlow* | 0.05 | 0.06 | 0.06 |
| GCN | Vaccinated NETTACK | 0.64 | 0.59 | 0.62 |
| GCN | Vaccinated *LowBlow* | 0.31 | 0.35 | 0.38 |
| t-PINE | Clean | 0.74 | 0.68 | 0.87 |
| t-PINE | NETTACK | 0.72 | 0.64 | 0.30 |
| t-PINE | *LowBlow* | 0.55 | 0.48 | 0.33 |
| t-PINE | Vaccinated NETTACK | 0.73 | 0.65 | 0.52 |
| t-PINE | Vaccinated *LowBlow* | 0.29 | 0.27 | 0.48 |

Table 2.4: Results overview. Comparison of poisoning and vaccination of GCN and t-PINE against NETTACK and *LowBlow*

$d = 8$, vaccinating t-PINE against *LowBlow* has no significant impact on the performance of t-PINE.

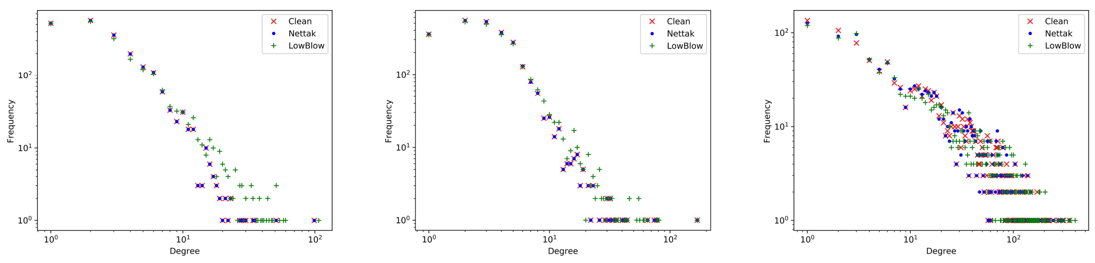
Due to the low-rank nature of *LowBlow*, it is more difficult to defend compared to NETTACK, and our vaccination method performs better on NETTACK rather than *LowBlow*.

### 2.5.5 Degree Distributions After *LowBlow*

In the previous subsection we demonstrated the effectiveness of *LowBlow* in fooling our proposed low-rank vaccination scheme, and deteriorating the performance of both GCN and t-PINE. In addition to the effectiveness of the attack, another important aspect that we would like to study experimentally is the effect of *LowBlow* in "what the graph looks like". In computer vision attacks, "look" can be easily defined by how a human perceives the poisoned data point/image. In graphs, however, such an intuitive metric does not exist.

Instead, [288] studies a proxy, which is the node degree distribution and how it is affected by the attack. In [288], the attack only affects one or a few nodes at a time, and thus, the attack results in a statistically insignificant alteration of the degree distribution.

*LowBlow*, on the other hand, by virtue of mixing the attack in high-valued singular components of the graph, this mixing may affect a number of nodes, resulting in statistically significant differences in the distributions, *for some nodes*.



(a) CiteSeer (b) Cora-ML (c) PoliticalBlogs

Figure 2.8: Degree distributions of the clean and attacked graphs on log-log scale. *LowBlow* affects the degree distribution only for the high-degree nodes, while leaving the majority of the nodes intact.

In Figure 2.8 we plot the degree distributions of the three real-world graphs we use, before and after the attack, in log-log scale. What we uniformly observe is that only the very high-degree nodes are affected by the attack, while the low and mid-degree nodes, which constitute the vast majority of this heavy-tailed distribution, remain intact, as far as their degree distribution is concerned.

To evaluate whether *LowBlow* perturbations are unnoticeable, we perform a statistical two-sample test for power-law distribution [288, 45] to see if the adjacency matrix after *LowBlow* perturbations follows similar degree distribution as the input graph. The null hypothesis $H_0$ proposes that the two samples have similar power-law distributions. Here, we compute the probability of not rejecting the null hypothesis where the two samples are from different distributions (*Type II error*). Similar to [288], we set the $p$-value to 0.95 which is a very conservative threshold and two samples from the same distribution are rejected 95% of the time. Following this conservative test, degree sequence of the graph after *LowBlow* perturbations does not follow the same power-law distribution as the input graph, i.e. the proposed low-rank attack is noticeable. To make the attack unnoticeable, we only consider edges that if added or removed from the graph, degree distribution will not change. After this step, we plotted the singular values of the graph before and after the attack and observed that the singular values are mainly different in higher ranks and the behavior is similar to NETTACK. This implies that an unnoticeable perturbation affects high-frequency spectrum of the graph. Consequently, our proposed vaccination mechanism successfully defends against unnoticeable adversarial attacks.

# Part II

# Image Classification

Chapter 3

# Defense Against Adversarial Attacks on Images

Manuscript is available on arXiv [95].

## 3.1 Introduction

In the last few years, Deep Neural Networks (DNNs) have been tremendously popular in various domains, including image processing and computer vision [131, 243, 254]. However, recently, the robustness of DNNs has been questioned when facing adversarial inputs. The performance of DNNs can significantly drop even on slightly perturbed instances [255]. For the task of image classification, attackers put constraints on perturbations such that they remain unnoticeable to the human eye, but they are still able to greatly deteriorate the performance of the model [222, 118, 28].

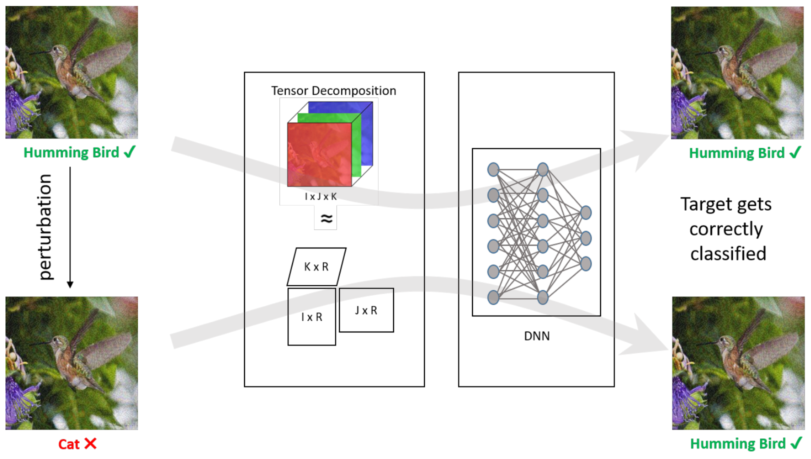


Figure 3.1: System Overview: low-rank tensor approximation of images to “vaccinate" the network against perturbations. (the term “vaccinate” was first used by Das et al. [77] to refer to models equipped with a defense mechanism.)

Utilizing machine learning methods that are vulnerable to adversarial attacks in a system where safety and security are critical factors may cause serious problems. Therefore, it is crucial to have a robust model against adversaries, especially in security-sensitive domains like autonomous driving and medical imaging. To address this concern, recent studies have researched to analyze the vulnerability of deep learning methods to come up with defense techniques against adversarial attacks [77, 49, 193, 223, 271].

To measure the strength of a perturbation, usually an $l_2$ or $l_\infty$ norm is used. Adversarial perturbations are mostly designed to have a small norm and are unnoticeable to human inspection [184]. Designing a defense mechanism is a difficult task. Typically, the defender has only access to the perturbed instances (and definitely not the original ones, where there would be hope to identify which parts have been tampered with) and should be able to defend against different types of perturbations. Moreover, a defense mechanism that is specialized against a particular kind of attack could be easily defeated by new attacks which are optimized against its strategy. Therefore, designing a defense technique that captures a universal pattern across various attacks is highly desirable since this will allow to defend against most of the adversarial attacks.

SHIELD, proposed by Das et al. [77], is a real-time defense framework that performs JPEG compression with random levels over local patches of images to eliminate imperceptible perturbations, which mostly appear in the high-frequency spectrum of images. In this chapter, we propose a tensor decomposition approach to compute a low-rank approximation of images that significantly discards high-rank perturbations. However, SHIELD considers images in isolation and ignores the correlation of images when facing adversarial attacks.

Our contributions are as follows:

- **Defense through the lens of factorization**: We propose a novel defense against adversarial attacks on images that utilizes tensor decomposition to reconstruct a low-approximation of perturbed images before feeding them to the deep network for classification. Without any retraining of the model, our method can significantly mitigate adversarial attacks.
- **Efficient and effective method**: Representing images with tensor allows processing images in batches as a 4-mode tensor, which is able to capture the latent structure of perturbations from multiple images rather than a single image and leads to more performance improvements.

## 3.2 Related Work

### 3.2.1 Adversarial Attacks

In this chapter, we focus on defending against adversarial attacks on deep learning methods for the task of image classification. Here, we briefly outline some of the most popular adversarial attacks on images.

Given a classifier $C$, the goal of an adversarial attack is to modify an instance $x$ to a perturbed instance $x'$ such that $C(x) \neq C(x')$, while keeping the distance $\|x - x'\|$ between perturbed and clean instance small [276]. By $\|.\|$ we denote some norm which is also used to express the strength of the perturbations. The popular choices are Euclidean distance ($l_2$ norm) and Chebyshev distance ($l_\infty$ norm). Here, we discuss some of the popular attacks against which we evaluate our proposed method.

**Fast Gradient Sign Method (FGSM)**[118]: FGSM is a fast method to compute perturbations that is based on computing first-order gradients. FGSM generates adversarial images by introducing a perturbation as follows:

$$x' = x + \epsilon \ \mathrm{sgn}(\nabla J_x(\theta, x, y)) \tag{3.1}$$

where $\epsilon$ is a user-defined threshold that determines the strength of the perturbations and controls the magnitude of perturbations per pixel. $\theta$ is the parameter of the model, $y$ is the true label of the instance $x$, and $J$ is the cost of training the neural network.

**Iterative Fast Gradient Sign Method (I-FGSM)**[169]: I-FGSM is the iterative version of the FGSM. In each iteration $i$, I-FGSM clips the pixel values to remain within the $l_\infty$ neighborhood of the corresponding values from a “clean” instance $x$:

$$x'_i = x'_{i-1} + \alpha \ \mathrm{sgn}(\nabla J_{x'_{i-1}}(\theta, x'_{i-1}, y)) \tag{3.2}$$

**Projected Gradient Descent (PGD)**[186]: PGD is one of the strongest gradient-based attacks [186] Given a clean image $x$, PGD aims to find a small perturbation $\delta \in \mathcal{S}$ to generate the perturbed instance $x' = x + \delta$. PGD starts from a random perturbation and iteratively updates the perturbation:

$$\delta_i = \Pi_{\mathcal{S}}[\delta_{i-1} + \tau \ \mathrm{sgn}(\nabla_x L(x + \delta_{i-1}, y))] \tag{3.3}$$

where $\tau$ is a fixed step size. $\Pi_{\mathcal{S}}$ projects the perturbation onto set $\mathcal{S}$, set of allowed perturbation in the $\epsilon$ neighborhood the “clean" instance $x$.

### 3.2.2 Defense Against Adversarial Attacks

Defense mechanisms against adversarial attacks fall into two main categories [276]. The first group of methods aim to train the classifier on adversarial examples to make it robust against them [142, 118, 263]. This approach requires extensive training on various adversarial examples. The second group of defenses try to remove perturbations from the adversarial example by performing some transformation [77, 123, 49, 123, 89]. These transformed examples are then fed to the model, and the objective is to achieve the same classification as the clean instance. In this chapter, our focus is on the second approach.

Shield proposed by Das et al. [77], uses image preprocessing as a defense mechanism to reduce the effect of perturbations. Shield is based on the observation that the attacks described above are high-frequency. Thus, eliminating those high-frequencies (which are not generally visible by the human eye) will sanitize the image. SHIELD performs Stochastic Local Quantization (SLQ) as a preprocessing step and subsequently employs JPEG compression with qualities 20, 40, 60, and 80 on the image, then for each $8 \times 8$ block of the image, randomly selects from one of the compressed images. SHIELD also retrains the model on images compressed with different JPEG qualities and uses an ensemble of these models to defends against adversarial attacks.

Variational Autoencoders (VAEs) [154, 86] have been used to defend against adversaries on images. In a study by Luo et. al. [185], they utilize VAE to map high-dimensional images to a lower-dimensional latent space in which most of the perturbation is discarded. Autoencoders [40] are a type of neural networks that learns a low-dimensional representation of the data in an unsupervised manner and variational autoencoders incorporate random sampling in the encoding and decoding process. The low-rank representation of images learned by the autoencoder can improve the performance of the deep model when facing adversarial examples. However, the drawback of autoencoder defenses is that they need to be trained on the data, which makes them not a suitable option for online applications.

In this Chapter, we preprocess images using tensor decomposition techniques to achieve a low-rank approximation of the image. We can significantly alleviate the effect of perturbations without performing any training on the dataset.

## 3.3 Proposed Method

In this section, we first investigate the characteristics of adversarial attacks on networks designed for the task of image classification. Then we propose a tensor-based defense mechanism against these attacks, which improves the performance of the network.

### 3.3.1 Characteristics of Image Perturbations

Assume a trained model $C$ with high accuracy on clean images is given. Adversarial attacks perform perturbations on the clean images in a way that they are imperceptible to humans yet are successful in deceiving the model to misclassify the perturbed instances. In other words, for a clean image $x$ and its corresponding perturbed image $x'$, the goal is to have: $C(x) \neq C(x')$. The adversarial attacks do not preserve the spectral characteristics of images and add high-frequency components to images to remain unnoticeable to the human eyes [77]. Perturbations in the image domain are crafted in a way that mostly affect the high-frequency spectrum of images. Therefore, discarding the high-frequency factors of the image using approaches like compression or low-rank approximation of images could be successful defense mechanisms against these types of perturbations. Therefore, a mechanism that only keeps the low-rank components of the image and discards the high-rank ones can be successful in discarding the perturbations. In [77], the authors leverage JPEG compression to remove high-frequency components of the image and alleviate the effect of perturbations. In this chapter, we study the problem from a "matrix spectrum" point of view (i.e., the singular value profile and the intrinsic low-rank dimensionality of the data) and use tensor decomposition techniques to achieve a low-rank approximation of perturbed images.

### 3.3.2 *TensorShield*: Tensor-based Defense Mechanism

In this section, we briefly the concepts and notations used in the chapter.

A tensor, denoted by $\underline{\mathbf{X}}$, is a multidimensional matrix. The order of a tensor is the number of modes/ways and is the number of indices required to index the tensor [160, 217]. An RGB image is a three-mode tensor where the first and second modes correspond to the pixels and the third mode corresponds to the red, green, and blue channels, i.e. the frontal slices are red, green, and blue channels of the image. An RGB image of size $W \times H$ is a 3-mode tensor of size $W \times H \times 3$, where $W$ and $H$ are the width and height of the image, respectively.

To achieve a low-rank approximation of the perturbed images, we perform a tensor decomposition technique on the image and by choosing small values for the rank of the tensor, we reconstruct a low-rank approximation of the image which is fed to the deep network. The low-rank approximation of image discards high-frequency perturbations which can improve the performance of the network on the perturbed images. However, traditional tensor decomposition techniques like CP/Parafac [129] and Tucker[265] are time-consuming and may slow down the neural network performance which makes our proposed method impractical for real-time defense. To overcome this issue, we leverage Tensor-Train decomposition [208] which scales linearly with respect to the dimension of the tensor and was especially introduced to address the problem of curse of dimensionality [208]. This highly-desirable property of the Tensor-Train allows us to process images in batches that form a 4-mode tensor and perform the Tensor-Train decomposition on 4-mode tensors quite fast. For a batch of $N$ images, the size of the 4-mode tensor will be $N \times W \times H \times 3$. Generally, decomposing

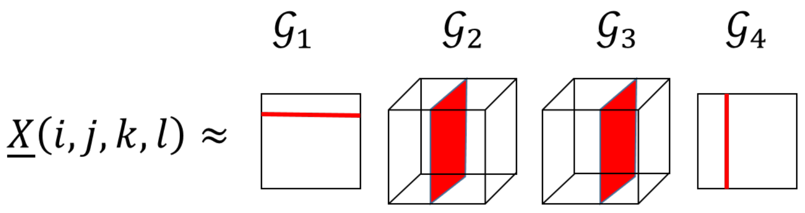


Figure 3.2: Tensor-Train decomposition of a 4-mode tensor.

a 4-mode tensor is slower compared to a 3-mode one. However, by considering images in batches, some of the I/O overhead is reduced, which results in almost the same processing time on the entire dataset. Furthermore, processing images in batches improves the performance of the model. The reason behind this is that decomposing images in batches extracts latent structure corresponding to perturbations from multiple images and captures general characteristics of perturbations.

For a 4-mode tensor, the Tensor-Train decomposition can be written as follows:

$$\underline{\mathbf{X}}(i,j,k,l) \approx \sum_{r_1,r_2,r_3} \mathbf{G}_1(i,r_1)\underline{\mathbf{G}}_2(r_1,j,r_2)\underline{\mathbf{G}}_3(r_2,k,r_3)\mathbf{G}_4(r_3,l) \tag{3.4}$$

Figure 3.2 illustrates the Tensor-Train decomposition of a 4-mode tensor.

Another possible representation for the batch of images is to convert the 4-mode tensor to a 3-mode tensor by stacking the images along the third mode, i.e., stacking RGB channels and the result tensor will be of dimension $W \times H \times 3 * N$. Figure 3.3 illustrates a 3-mode stacked tensor of $N$ images. There are other ways to convert a 4-mode tensor into a 3-mode one. For instance, another way is to flatten the RGB image into a matrix with three columns corresponding to the channels of the image. With this representation, the final tensor will be of size $W * H \times 3 \times N$. One disadvantage of this representation is that flattening the image ignores the spatial relationship of the pixels. Moreover, with this vectorized representation, the first dimension is much bigger than the other two dimensions and requires a larger value of rank to get a reasonable approximation of the image, and larger ranks make the decomposition slower. For these reasons, we do not consider the vectorized representation in our study. In the experimental evaluations that follow, we will examine different representation including a single image versus batches of images and 3-mode tensors versus 4-mode tensors.

## 3.4 Experimental Evaluation

In this section, we show how the proposed method can successfully remove adversarial perturbations and we compare our results to Shield (SLQ). According to [73], original Shield evaluations have gained benefit from central cropping of images in their evaluations, whereas the perturbations were generated with cropping being off. In all our evaluations, we disable the central cropping.

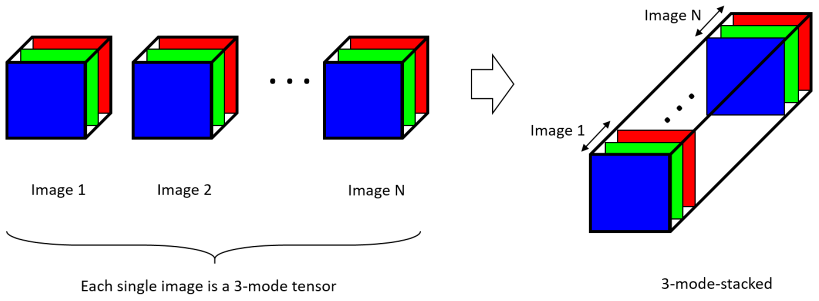


Figure 3.3: Stacking 3-mode images along the third mode.

### 3.4.1 Experiment Setup

We performed experiments on the validation set of the *ImageNet* dataset, which includes 50,000 images from 1,000 classes [84]. All experiments are performed on the *ResNet-v2 50* model [131] from the *TF-Slim* module of *TensorFlow* [15]. The adversarial attacks are from the CleverHans package [1] [220]. We performed the experiments on a machine with one NVIDIA Titan Xp (12 GB) GPU. We used TensorLy [2] library in Python to perform tensor decomposition techniques [163].

### 3.4.2 Parameter Tuning

In our evaluations, we express different configurations in the form of a list as [tensor decomposition, tensor representation, batch size, rank] and we investigate the accuracy and runtime of the ResNet-v2 50 on 1000 images from the ImageNet dataset for different configurations. The possible values for each part of the configuration list is as follows:

- Tensor decomposition: {Parafac, Tucker, Tensor-Train}
- Tensor representation: {3-mode, 3-mode-stacked, 4-mode}
- Batch size: {1, 5, 10, 20, 50}
- Rank: varies by choice of tensor representation and decomposition.

Performing tensor decomposition for a batch of images can reduce the decomposition overhead compared to decomposing a single image and accelerates the entire evaluation process. Moreover, considering images in batches helps to better capture the pattern of perturbations from multiple images. However, the choice of the right batch size is important.

[1]https://github.com/tensorflow/cleverhans
[2]https://github.com/tensorly/tensorly

A large batch of images needs larger ranks for decomposition and could get very slow. Also, in a large batch of images, the variety of images that are from different classes increases which deteriorates the performance of the decomposition. To find the best batch size, we perform a grid search on values 5, 10, 20, and 50. Tensor Train decomposition of a 4-mode tensor requires setting 3 values for the ranks. The first value corresponds to compressing the batches, the second value corresponds to compressing the image pixels, and the third value corresponds to compressing the RGB channels. We fix the first rank to the number of batches and the third rank to the number of channels i.e., 3. For the second rank, we search within the range 40 to 150. Figure 3.4 shows the accuracy and runtime of the model for different batch sizes for Tensor-Train decomposition with ranks ranging from 50 to 120 with steps of 5. The figure also shows how processing single images (batch size 1) differs from batch sizes greater than 5. In the case that we are processing single images, the runtime increases as the rank gets larger, however, as the batch size increase, the runtime becomes less sensitive to the ranks and for the batch size 50 it will become almost constant for all the ranks. Batch size 5 produces the highest accuracy, while batch size 10 has the lowest runtime. There is a trade-off between runtime and accuracy. Based on the priorities of the system, one might sacrifice accuracy for speed.

Figure 3.5 shows the effect of different batch sizes on the 3-mode-stacked representation. Plots for batch sizes 5, 10, and 20 are almost identical in both accuracy and runtime. Batch size 50 produces the curacy with the 3-mode-stacked representation. However, the highest accuracy with the 3-mode-stacked representation is lower than the highest accuracy achieved using the 4-mode representation.

### 3.4.3 What If the Attack Is Low-Rank?

What happens if the attacker learns about our low-rank approximation defense and attempts to generate perturbations which affect the lower end of the singular value spectrum of the images? Here, we explore this question and show that crafting such per-

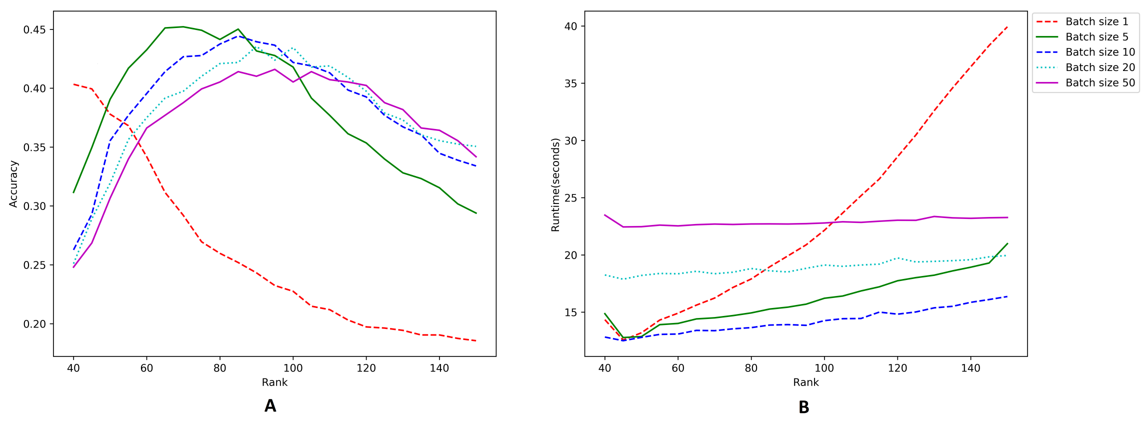


Figure 3.4: Accuracy (A) and runtime (B) of ResNet-v2 50 over 1000 images attacked by FGSM ($\epsilon = 4$). Tensor-Train decomposition is applied on a single image (batch size 1) or 4-mode tensor of batches of size 5, 10, 20, and 50 to defend against FGSM perturbations.

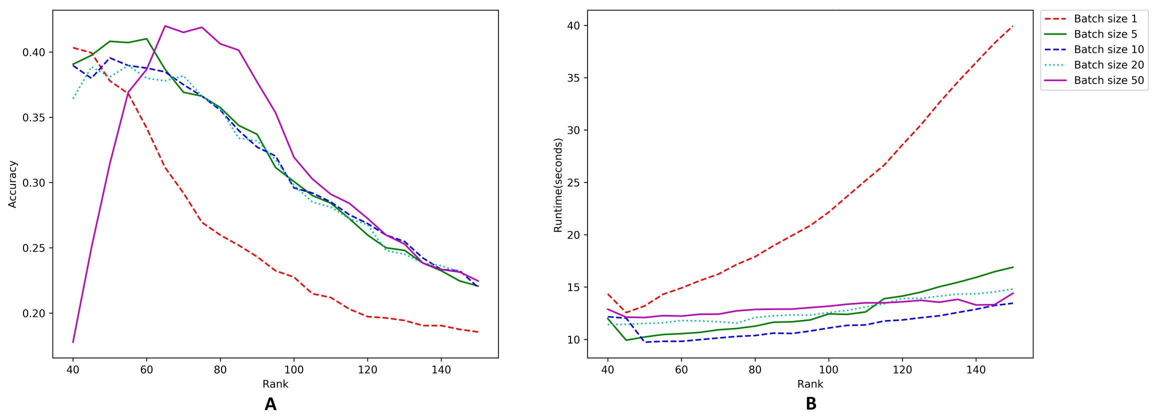


Figure 3.5: Accuracy (A) and runtime (B) of ResNet-v2 50 over 1000 images attacked by FGSM ($\epsilon = 4$). Tensor-Train decomposition is applied on a single image (batch size 1) or 3-mode-stacked tensor of batches of size 5, 10, 20, and 50 to defend against FGSM perturbations.

turbations creates noisy, low-quality images that are no longer indistinguishable from the original images.

Our low-rank tensor-base defense mechanism relies on the intuition that image perturbations mostly affect the high-frequency spectrum. This is because attackers aim to craft perturbations that are unnoticeable to human eyes. To confirm this intuition, we manipulated FGSM image perturbations to generate low-rank adversarial examples. To generate a low-rank attack, we follow the steps mentioned in [93] to replace significant singular components of FGSM perturbations within the range [20,30) with the corresponding components of the clean image. Figure 3.6 shows images perturbed using the FGSM attack versus the low-rank attack. Low-rank perturbed images are obviously noisy and pixelated which are far from the objective of unnoticeable perturbations.

### 3.4.4 Results

As mentioned in Section 3.3, Tensor-Train performs much faster than Parafac and Tucker. Therefore, for the Parafac and Tucker, we only report the result for the configuration which corresponds to the maximum accuracy as a reference for comparison against Tensor-Train.

#### Comparing Against SLQ on ImageNet

We compare the performance of *ResNet-v2 50* model when vaccinated using SLQ and *TensorShield* methods. PGD, FGSM, and I-FGSM attacks are used to generate adver-

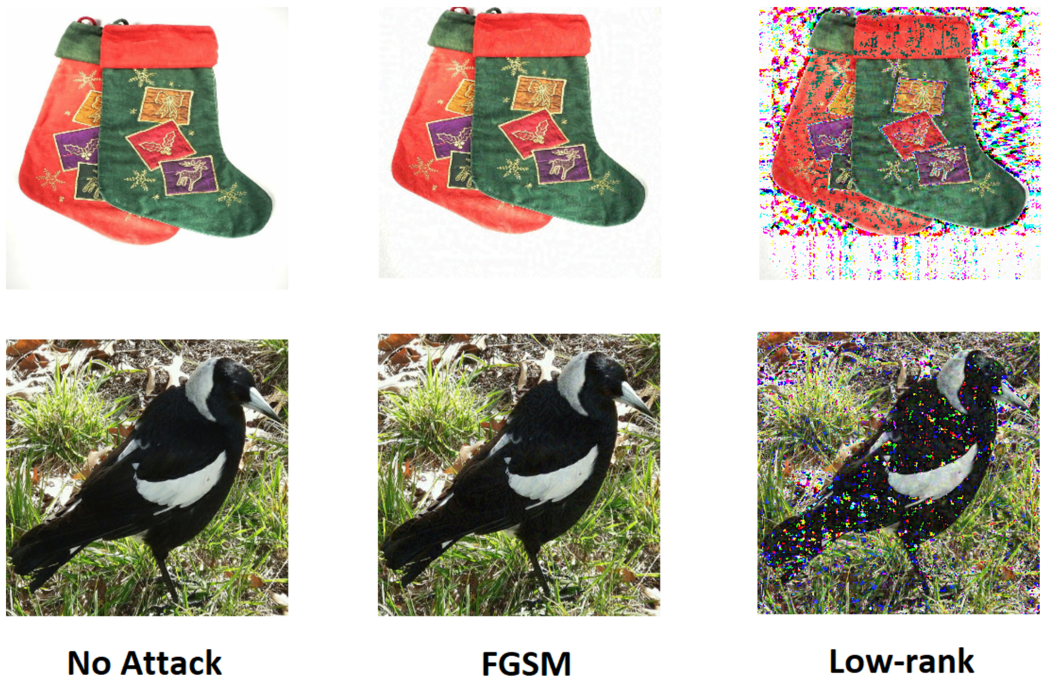


Figure 3.6: Examples of low-rank perturbation on images. Perturbations that mostly affect high-frequency spectrum of images are not visible to human eyes, however, low-rank perturbations generate noisy and pixelated images.

sarial examples. Table 3.1 summarizes the result. Different configurations of *TensorShield* are reported in the table.

| **Configurations** | **PGD** ($\epsilon = 4$) | **FGSM** ($\epsilon = 4$) | **I-FGSM** ($\epsilon = 4$) | **Runtime** (seconds) |
|---|---|---|---|---|
| No defense | 11.10 | 18.40 | 7.49 | |
| [Tensor-Train, 4-mode, 5, [5,90,3]] | **51.53** | **43.59** | **50.46** | 675 |
| [Tensor-Train, 4-mode, 10, [10,100,3]] | 51.01 | 43.10 | 49.95 | 605 |
| [Tensor-Train, 3-mode, 1, 40] | 49.75 | 42.32 | 48.52 | **530** |
| [Tucker, 3-mode-stacked, 30, [105,105,90]] | 49.37 | 40.07 | 48.79 | 1050 |
| [Parafac, 3-mode, 1, 60] | 48.11 | 41.38 | 49.75 | 5500 |
| SLQ | 44.60 | 29.40 | 38.60 | **410** |

Table 3.1: Summary of accuracies and runtime of ResNet-v2 50 on ImageNet validation set against FGSM, I-FGSM, and PGD adversarial attacks for defenses with different configurations.

As illustrated in Table 3.1, Tensor-Train outperforms Tucker and Parafac with respect to both accuracy and runtime. Tensor-Train performed on 4-mode tensor has produced the highest accuracy. As explained earlier, processing images in batches better captures latent components corresponding to perturbation by leveraging higher-order correlations. Tensor-Train can be utilized with different tensor representations (3-mode, 3-mode-stacked, or 4-mode) to adjust to needs for higher accuracy or higher speed. While the 4-mode representation produces the highest accuracy, the 3-mode single image representation can be used to speed up the process, with a small drop in the accuracy. SLQ is the fastest among all defenses, but it has the lowest accuracy.

**Comparing Against VAE on MNIST**

Here, we compare our method against the variational autoencoder defense proposed in [185] on the MNIST dataset. Defense mechanisms based on variational autoencoders, require extensive training and hence they are not easily scalable to larger datasets. Therefore, we compared our method against VAE defense on the MNIST dataset which is a small dataset. CNN model used for the image classification has the same architecture as explained in [185]. The CNN model contains 2 convolutional layers followed by a fully-connected layer. Both of the convolutional layers use $5 \times 5$ filter size with 32 and 64 channels in the first and second convolutional layers, respectively. After each convolutional layer, $2 \times 2$ max-pooling with stride 2 is used. The fully-connected layer has 1024 hidden nodes that uses Relu activation with a dropout rate of 0.4.

| **Defense** | **No attack** | **PGD** | **FGSM** | **I-FGSM** |
|---|---|---|---|---|
| | | $(\epsilon = 4)$ | $(\epsilon = 4)$ | $(\epsilon = 4)$ |
| No defense | 99.12 | 97.85 | 97.80 | 97.50 |
| JPEG-20 | 99.12 | 98.28 | 98.13 | 98.20 |
| JPEG-40 | 99.12 | 98.35 | 98.20 | 98.27 |
| JPEG-60 | 99.12 | 98.29 | 98.17 | 98.28 |
| JPEG-80 | 98.11 | 98.25 | 98.09 | 98.10 |
| VAE | 98.87 | 98.30 | 98.23 | 98.18 |
| *TensorShield* | 99.12 | **98.44** | **98.32** | **98.31** |

Table 3.2: Summary of accuracies of CNN model on MNIST dataset against PGD, FGSM, and I-FGSM adversarial attacks. The performance of the model is reported when it is vaccinated using different defenses including *TensorShield*, VAE, and JPEG compression with qualities 20, 40, 60, and 80.

Table 3.2 summarizes the performance of different defenses including *TensorShield*, VAE, and JPEG compression with qualities 20,40,60, and 80. For *TensorShield* we only report the configuration with the best accuracy which is a 3-mode tensor containing batches of 5 images $(28 \times 28 \times 5)$. We used Tensor-Train decomposition with rank 30. Size of MNIST images is $28 \times 28$ and a tensor including a single image will be a 2-dimensional array and a tensor composing a batch of $N$ images will be a 3-mode tensor of size $28 \times 28 \times N$.

Figure 3.7 shows the performance of *TensorShield* and VAE against the FGSM attack. Also different quality of JPEG compression was applied on images as a preprocessing step to resist against adversarial attacks. For small perturbations that are less noticeable to human eyes, *TensorShield* is on par or better than VAE, but *TensorShield* has a higher drop in performance for stronger attacks. Even for strong attacks, the performance improvement achieved by *TensorShield* is significant and due to its scalability, *TensorShield* is a feasible option for vaccinating models on large datasets.

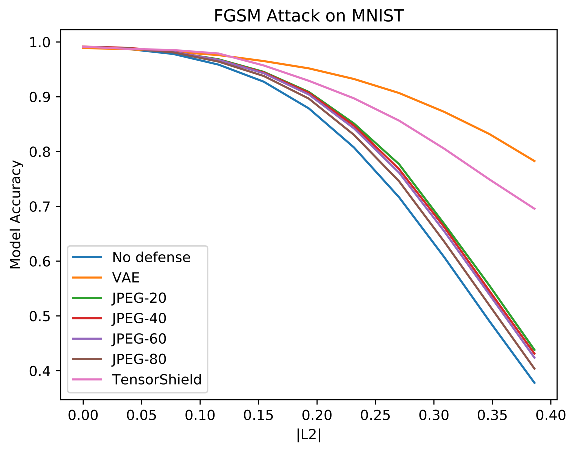


Figure 3.7: FGSM attack on CNN model for MNIST dataset. For small and less noticeable perturbations, *TensorShield* outperforms other defenses, however as perturbations get stronger, VAE outperforms our method. However, VAE requires extensive training which makes it less applicable to large datasets, whereas *TensorShield* does not require any training on the dataset and can be applied on large datasets.

**Introducing Randomness to the Defense Framework**

Incorporating randomness in the defense framework makes the job of the attacker more difficult to deal with a random strategy rather than a fixed one. By selecting randomly from a set of ranks, we can add randomness to the tensor decomposition process. Another way is to split images into small patches, similar to local $8 \times 8$ patches from SHIELD, and perform decomposition of a random rank on each patch and stitch up the patches to reconstruct a randomized low-rank approximation of images. In a 4-mode tensor representation, splitting images into patches creates smaller 4-mode tensors, e.g. splitting a 4-mode tensor including 5 batches of images with size $300 \times 300 \times 3$ into patches of size $50 \times 50$ creates 6 tensors of size $5, 50, 50, 3$. Table 3.3 shows the results of incorporating randomness with tensor decomposition. With smaller patch sizes, the decomposition will have more overhead, and hence the runtime complexity increases.

| **Patch size** | **Ranks** | **PGD** | **FGSM** | **I-FGSM** | **Runtime** |
|---|---|---|---|---|---|
| | | ($\epsilon = 4$) | ($\epsilon = 4$) | ($\epsilon = 4$) | (seconds) |
| [8, 8] | {[5,3,3,[5,5,3],[5,7,3]} | 47.89 | 41.04 | 48.36 | 2550 |
| [50,50] | {[5,10,3],[5,20,3],[5,30,3]} | 48.35 | 41.97 | 48.12 | 1100 |
| [150,150] | {[5,40,3],[5,50,3],[5,60,3],[5,70,3]} | 50.96 | 42.12 | 48.98 | 765 |
| No patching | {[5,70,3],[5,90,3],[5,110,3]} | 50.48 | 42.73 | 49.68 | 710 |

Table 3.3: Accuracies and runtime of ResNet-v2 50 on ImageNet validation set against PGD, FGSM, and I-FGSM adversarial attacks with $\epsilon = 4$ vaccinated using Tensor-Train with 4-mode tensor of batch size 5. Decomposition rank is randomly selected from a set of possible ranks. No patching is equivalent to full size image.

# Part III

# Recommender Systems

# Chapter 4

# Defense Against Adversarial Attacks on Recommender Systems

## 4.1 Introduction

Recommender systems are powerful tools to help users better and easier choose between millions of options in different scenarios such as shopping and movie, book, or song selection. Business owners also gain benefits from appropriate recommendations that can potentially increase their revenue. Collaborative filtering techniques are widely used in recommender systems due to their simplicity and strong performance. Collaborative filtering relies on the user-item interactions in the past to predict future interactions. The idea is that users with similar choices in the past are most likely to make similar choices in the future [235]. Matrix factorization is a popular collaborative filtering technique that considers user-item interactions and tries to learn latent factors for users and items [162, 161]. Recently, neural collaborative filtering models have also been utilized to model non-linear user-item interaction [133].

Recent studies in adversarial machine learning show that machine learning algorithms are susceptible to adversarial attacks [76, 118, 255], and recommender system models are also susceptible to adversaries[83]. Attackers aim to fool recommender systems to recommend products (or items, in general) to users that serve their malicious intentions rather than satisfying users' needs. Attackers may achieve their objective by writing fake product reviews, creating fake user profiles, manipulating product images, etc. It is crucial to detect these adversaries and make recommender models robust against them. In this chapter, we investigate characteristics of two of the adversarial attacks on recommender systems proposed by Tang et al. [256] and Christakopoulou et al. [71]. Then we report our experimental evaluations on how to make the recommender system more robust against these types of attack.

## 4.2 Related Work

### 4.2.1 Adversarial Attacks on Recommender Systems

Recommender systems have always been a target of attackers. Traditionally attackers tried to inject hand-engineered fake profiles to affect the recommender system to offer their target items maliciously. Injecting fake user profiles is broadly called a shilling attack. The shilling attack aims to augment some user profiles with limited item ratings. They can diverge the recommendation result and force the recommender system to recommend some target items to users. In recent years, poisoning attacks leveraged machine learning algorithms to generate fake user profiles [177, 109, 71]. Li et al. [177] proposed a data poisoning attack on factorization-based recommender systems. In a recent work by Christakopoulou et al. [71], Generative Adversarial Network (GAN) [117] is used to generate fake user profiles with similar rating distribution of real users. This ensures that the fake users generated by GAN are unnoticeable. Then by applying iterative gradient descent, a final set of fake user profiles are generated and injected into the recommender system.

### 4.2.2 Defense Against Adversarial Attacks on Recommender Systems

Adversarial training is the most popular approach to make recommender systems robust against adversaries [257]. In adversarial training, adversarial examples are generated using an existing attack model, and then these adversarial examples are fed to the machine learning model along with the benign examples. Machine learning models trained on the adversarial instances will have a better performance on the adversarial examples and higher generalization power. Adversarial training has been applied to recommender systems to have a more robust model [257, 269]. Adversarial training is expensive as it requires retraining of the model over benign and adversarial examples. Moreover, there are many attack models with different attack strategies and the adversarially-trained model may not work well to defend against unseen adversarial examples. Another group of defense techniques in the literature performs a preprocessing step to transform an adversarial example into a similar benign example. The goal of the transformation is to discard the adversarial artifacts. In the image domain, this can be done by removing the high-frequency noise added to the image [77, 95]. A similar idea was also applied to defend against attacks on graphs [93]. The idea behind these methods is that attackers try to generate unnoticeable perturbations, and therefore perturbations primarily affect the high-frequency domain of images and graphs. In this chapter, we explore the characteristics of adversarial attacks from [256] and [71] to see how a low-rank approximation approach can help to defend against adversarial attacks.

## 4.3 Proposed Method

In Section 4.3.1, we first briefly talk about two of the recent adversarial attacks. Next, in Section 4.3.2, we explain our proposed methods to improve the performance of the recommender system when adversarial user profiles are present.

### 4.3.1 Adversarial Attacks in Recommender Systems

In this section, we investigate the characteristics of two different algorithms to generate adversarial users to attack recommender systems. Below, we explain the threat model for each method.

#### Attack I - RecSys20

The first adversarial model is proposed by Tang et al. [256]. This method uses publicly available data used by recommender systems to learn about user preferences and generates fake user profiles to maliciously influence the recommender system. The malicious intention is to boost the chance of a target item being recommended to users. Using a surrogate model, fake users profiles are learned as a bi-level optimization problem. The inner objective is to minimize the loss of the surrogate model with the presence of the fake users and the outer objective is to minimize the loss of the adversarial model by maximizing the chance of recommending of target items to real users.

**Attack II - RecSys19**

The second method is proposed by Christakopoulou et al.[71] which generates fake users in two steps. Their proposed framework first uses deep convolutional generative adversarial network (DCGAN) [230] to generate initial fake user profiles. These generated user profiles have a similar rating distribution to the real users. Next, their model iteratively updates the fake user profiles to maximize the hit ratio of a target item(s). GAN generates realistic-looking user profiles that result in an unnoticeable attack. Figure 4.1 compares real vs. fake user profiles generated using GAN.

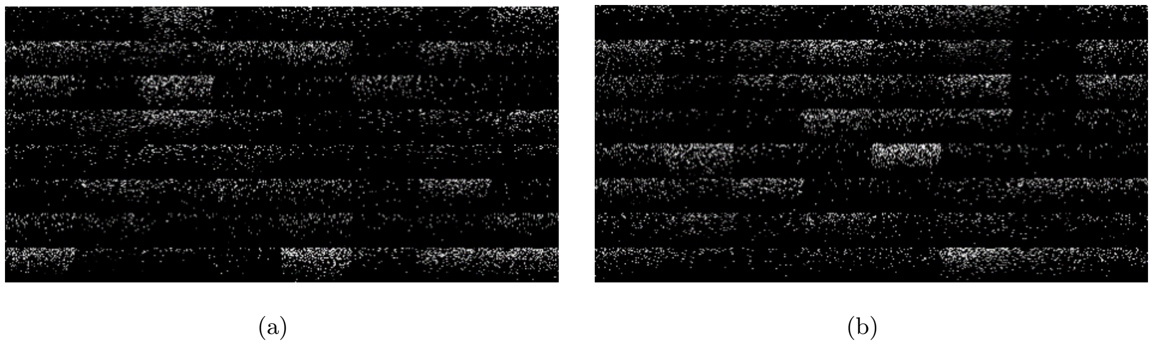


Figure 4.1: (a) Real user profiles vs. (b) fake user profiles learned by DCGAN for the MovieLens dataset. Black pixels indicate missing ratings while bright pixels indicate ratings for movies. The maximum rating is 5 that corresponds to white pixels.

In [71], the assumption is that the recommender system is oblivious to the existence of an adversary, therefore it optimizes its loss over all given user profiles, including the fake profiles.

Previous study on adversarial attacks on images and graphs [77, 93, 95] elaborated the fact that adversaries mainly impact high-frequency components of the data to remain unnoticeable. We are interested to examine if the same observation could be extended to adversarial recommender systems. In the next section, we propose different methods to diminish the negative impact of adversarial users.

### 4.3.2 Defense Methods

In this section, we introduce three low-rank defense methods to lessen the harm of fake users and improve the performance of the recommender system.

#### Fake User Detection

Our goal is to train a classifier on real and fake user profiles that can successfully detect fake users. Once fake users are detected, they can be removed from the input data fed to a recommender model to improve its performance. A good choice of a classifier is a model that is simple and can run very fast during to be applicable in online recommendation. Also, it should have high recall because false negatives are very costly and we do not want to misclassify fake users as real ones, whereas classifying some of the real users as fake (false positive) does not harm the recommendation. We use Support Vector Machine (SVM) model to classify fake and real users. The training data is highly imbalanced (131 fake users vs. 13.1k real users) and extremely sparse (99.7% sparsity). For such a high-dimensional data classifiers will have a poor performance. We use Principal Component Analysis (PCA) to reduce the dimensionality of the data and then train the classifier in the lower-dimensional space. A 2D visualization of data considering the first two principal components is illustrated in Figure 4.2. As shown in Figure 4.2, points corresponding to fake and real users are separable in the 2D space. A preferred model should detect fake users (true positives) with high confidence while the number of false negatives is low. A fake user misclassified as a real user can harm the recommender system, but detecting a real user as a fake once is safe. Therefore, a good model should have a high recall. Once a high-performing model is found, detected fake users by the model are removed from the dataset and recommender model is trained on the rest of the data.

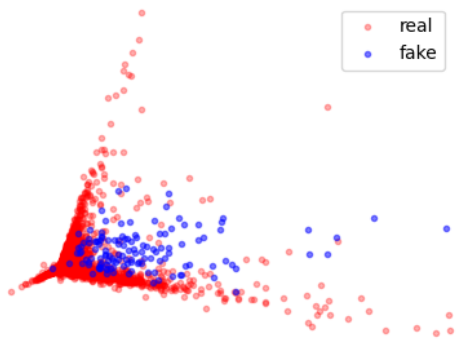


Figure 4.2: 2D visualization of Gowalla dataset and learned fake users using PCA transformation.A simple SVM model is able to detect fake users, shown in blue, from real users, depicted in red

### Local Low-rank Reconstruction

Despite the successful performance of fake user detection defense technique, it has some limitations. It is a supervised model and requires training instances and examples of fake user profiles to train the classifier. Therefore, for different threat models, we require to obtain fake user examples and train our model against them. It is not always feasible to consider every different attack model and there are various unknown adversaries. The goal of this section is to propose an unsupervised defense model that can be applied without prior knowledge about the adversarial examples.

The number of fake users generated by adversarial models is very small compared to the number of real users in the system. Adversarial model generates fake users that mimic the same behavior of real users to avoid being easily detected and remain unnoticeable. The fake users added to the system forces the recommender model to boost the hit ratio for the target items and the overall performance of the system is preserved for imperceptibility reasons. The impact of these fake users are very subtle compared to the impact of large number of real users and with similar intuition as [93], a low-rank solution is able to alleviate the negative impact of fake users. For a large user-rating matrix, performing a low-rank SVD requires a fairly large rank to capture main components of the data which makes the reconstruction slow. Also, reconstruction will fill the missing values and adds new values to the rating matrix. We perform local SVD low-rank reconstructions on small patches of the rating matrix and put the reconstructed patches back together to reconstruct the entire rating matrix. Reconstruction of small patches requires a very small SVD rank and can be done in parallel to speed up the process. In the experimental evaluation that follows, we consider two cases where adversarial users are absent or present and share how the performance of the recommender system is affected by the local low-rank reconstruction in both cases.

### Low-rank Transformation

Low-rank SVD reconstruction is able to alleviate the impact of fake users. However, we cannot infer that the performance improvement is due to the fact that the adversarial attacks are high-rank. Low-rank reconstruction introduces some additional ratings that could lead to performance improvements. To investigate this, we try to answer the following questions:

- What is the impact of low-rank SVD components?
- What is the impact of new rating values introduced from reconstruction?

To answer the first question, we transform latent factors of the matrix factorization model into SVD space. We perform the following steps:

Given $X_A$ which is the attacked ratings matrix:

$$X_{A_k} = U_k \Sigma_k V_k^T \tag{4.1}$$

where $P_a = U_k U_k'$ and $P_b = V_k V_k'$ are the transformation matrices. These $P_a$ and $P_b$ matrices are used to transform latent factors $A$ and $B$ into the SVD space:

$$A' = P_a A \quad and \quad B' = P_b B \tag{4.2}$$

Original latent factors $A$ and $B$ are replaced with the transformed latent factors $A'$ and $B'$. According to our experimental evaluations described in Section 4.4.2, transforming MF latent factors into SVD space improves the performance of the recommender system. This shows that the performance improvement we gain from SVD low-rank reconstruction is not all because of the new ratings, and SVD components are able to discard the adversarial components.

To answer the second question regarding the impact of new rating values, we are interested in comparing the impact of transformation vs. new rating values to see which one yields higher improvements. We gradually add new rating values to the original attacked profile matrix from 0% (no new value) to 100% (all of the new values considered). The experimental evaluations reveals that recommender systems gains a higher benefit when we perform latent factor transformation along with new ratings generated after low-rank reconstruction.

## 4.4 Experiments

We describe the dataset and experimental settings for our experiments and we share the results for each adversarial model described in Section 4.3.1.

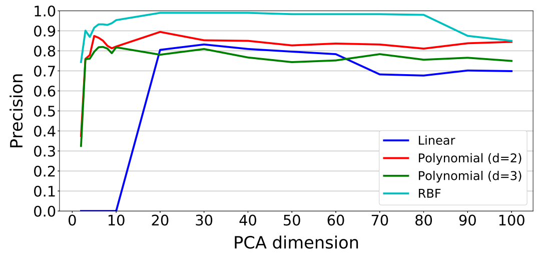
### 4.4.1 Attack I - RecSys20

#### Dataset and Experiment Setup

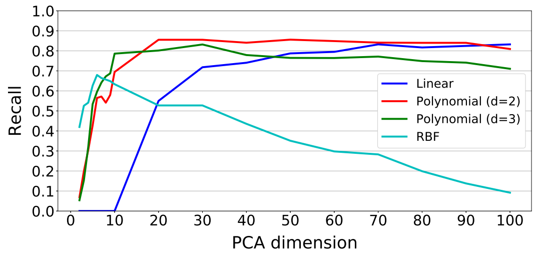
In this part of our experiments, we use Gowalla [69] dataset and we follow the same procedure as explained in [256] to preprocess the data. Gowalla dataset is an undirected location-based scocial network where users can share their locations. After preprocessing, the dataset has 13.1k users and 14.0k items. Data is randomly split into training and test set with ratio 80:20. Fake users are generated following the evaluation protocol explained in [256]. Target items are randomly selected from 4 different click percentiles:

- Head: items with total clicks greater than $95^{\text{th}}$ percentile.
- Upper torso: items with total clicks between 75 and 95 percentiles.
- Lower torso: items with total clicks between 50 and 75 percentiles.
- Tail: items with total clicks less than $50^{\text{th}}$ percentile.

In our experiments we only consider head target items and Weighted Regularized Matrix Factorization model [137] with stochastic gradient descent optimization (WRMF(SGD))



(a) Precision



(b) Recall

Figure 4.3: Performance of SVM classifer on the imbalanced data. SVM with second degree polynomial kernel achieves high recall with lower number of PCA components (n=20).

that achieves stronger adversarial performance according to the results shared in [256]. Number of fake users generated is 1% of real users, i.e. 131 fake users are generated to boost the hit ratio of the target items (HR@50).

**Fake User Detection**

Figure 4.3 shows the results of 10-fold cross-validation for the SVM model with different Kernels trained on various PCA dimensions. Also, we consider different cases where the training data is imbalanced or down-sampled. In the case of imbalanced data we use stratified cross-validation.

SVM model with second degree polynomial kernel achieved the highest recall at PCA with 20 components. Lower number of PCA components are preferable as it is faster to transform the data. We pick the SVM model with second degree polynomial to detect and remove fake users. The SVM classifier first scans all the user profiles and predicted fake users are removed from the data. Next, Recommender system recommends items to users using the sanitized data. The Performance of the WRMF(SGD) model with and without

| Defense | Data | Overall | | Target Items | |
|---|---|---|---|---|---|
| | | Recall@50 | HR@50 | Recall@50 | HR@50 |
| No Defense | Clean | 0.2895 | 0.7590 | 0.0207 | 0.1021 |
| | Attacked | 0.2884 | 0.7580 | 0.0261 | 0.1415 |
| 2nd degree polynomial SVM | Clean | 0.2887 | 0.7580 | 0.0203 | 0.1008 |
| | Attacked | 0.2898 | 0.7592 | 0.0252 | 0.1095 |

Table 4.1: Impact of the SVM fake user detection on the performance of WRMF(SGD) model on clean and attacked data from Gowalla dataset.

the presence of the fake user detector is reported in Table 4.1. In our experiments, we report the performance on the target items to show how the defense method reduces the chance of target items being recommended to users. In addition, we report the overall performance of the recommender system to show that the defense method does not adversely affect the overall performance of the recommendation model.

SVM model is able to detect about 85% of fake users and causes the target items HR@50 to drop about 4% which is very close to the performance of the recommender system on the clean data.

### Local Low-rank Reconstruction

In this section, we share the result for the impact of the low-rank reconstruction on the recommender system's performance. Table 4.2 summarizes the performance of the WRMF(SGD) model with low-rank SVD reconstruction to defend against the fake users. Parameters of the local SVD reconstruction is reported in the form of [patch size, SVD rank]. Patch size = x means the entire rating matrix is considered as a single patch. We report different settings of hyperparameters that helped to reduce the hit ratio on the target items while maintaining the overall performance of the recommender model.

### 4.4.2 Attack II: RecSys19

#### Dataset and Experiment Setup

We used the MovieLens 100k dataset [128] in our experiments. MovieLens 100K dataset contains ratings for 1682 movies from 943 users. Ratings are within range 1 - 5. The recommender system model in our experiments is probabilistic matrix factorization (PMF) [199]. We use metrics such as recall@K and precision@K to report the performance of the recommender system.

| Defense | Data | Overall | | Target Items | |
|---|---|---|---|---|---|
| | | Recall@50 | HR@50 | Recall@50 | HR@50 |
| No Defense | Clean | 0.2895 | 0.7590 | 0.0207 | 0.1021 |
| | Attacked | 0.2884 | 0.7580 | 0.0261 | 0.1415 |
| [100, 90] | Clean | 0.2893 | 0.7580 | 0.0241 | 0.1105 |
| | Attacked | 0.2889 | 0.7612 | 0.0260 | 0.1138 |
| [300, 80]] | Clean | 0.2855 | 0.7515 | 0.0216 | 0.0987 |
| | Attacked | 0.2838 | 0.7514 | 0.0253 | 0.1091 |
| [150, 20]] | Clean | 0.2609 | 0.7148 | 0.0238 | 0.1092 |
| | Attacked | 0.2601 | 0.7168 | 0.0236 | 0.1004 |
| [x, 3500] | Clean | 0.2500 | 0.7117 | 0.0209 | 0.0963 |
| | Attacked | 0.2468 | 0.7090 | 0.0236 | 0.1029 |
| [x, 5000] | Clean | 0.2866 | 0.7563 | 0.0231 | 0.1049 |
| | Attacked | 0.2866 | 0.7582 | 0.0278 | 0.1226 |

Table 4.2: Impact of the local SVD reconstruction on the performance of WRMF(SGD) model on clean and attacked data from Gowalla dataset.

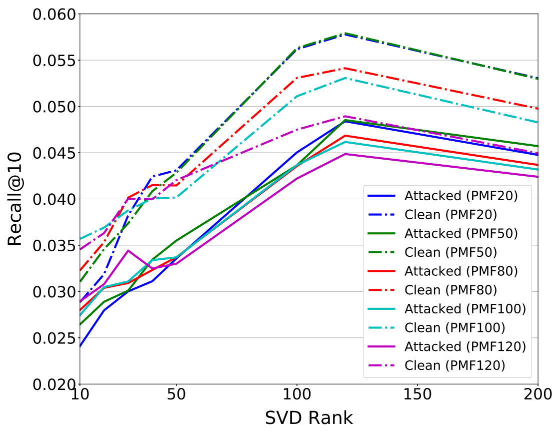


Figure 4.4: Performance of PMF recommender with low-rank reconstruction of clean and attacked user profiles.

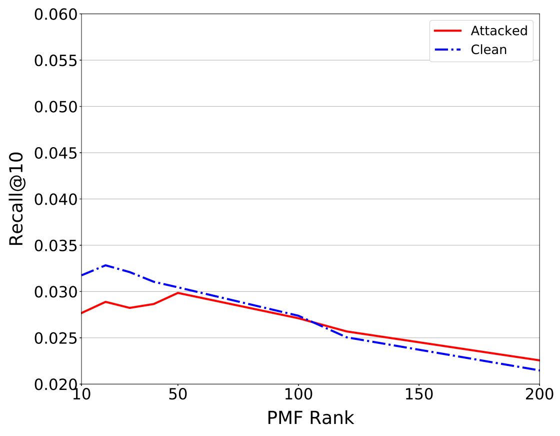


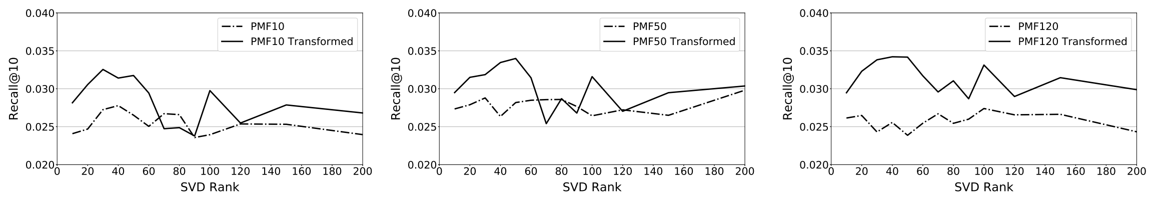
Figure 4.5: Impact of different PMF ranks on the performance of recommender system for clean and attacked data.

### Low-rank Transformation

In our experiments, We performed low-rank reconstruction of the attacked user profiles and fed it to the PMF model. Figure 4.4 illustrates how SVD low-rank reconstruction of the attacked profiles, improves the performance of the recommender system. We performed this experiment for different PMF ranks to evaluate the impact of PMF ranks on the performance of the recommender system and the result is shown in Figure 4.5. SVD low-rank reconstruction is applied with different ranks ranging from 10 to 200. Both clean and attacked user profiles gain benefits from low-rank reconstruction up to rank 120. SVD reconstruction using Ranks greater than 120 incorporates the adversarial components and leads to a performance drop.

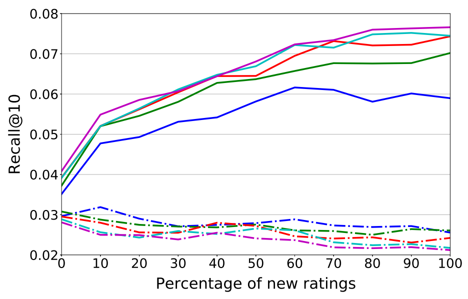
Moreover, to investigate if the performance improvement is because of the high-rank nature of the attack or due to the new rating values introduced after the reconstruction which where originally missing, we performed another set of experiments following the steps explained in Section 4.3.2. The performance of the recommender system after the transformation is shown in Figure 4.6 and we can observe that the recommender model benefits from the transformation without considering the new rating values.

In another experiment, we gradually added new ratings from SVD reconstruction together with transformation of latent factors. Figure 4.7 shows the results. 0% means that no new rating values is added and the performance is reported on the original attacked matrix. 100% means all of the new rating values is added to the ratings matrix. Dashed lines show the performance of the recommender before transformation while gradually introducing new rating values and the solid lines are the performance after the transformation and introducing the new rating values. These plots show that the improvement gained from transformation is much higher than what is gained from new ratings added. For SVD ranks 50 and 100, adding new values does not affect the performance of the recommender, but at SVD ranks 150 and 200, adding more data yields higher recall@10 before transformation. On the other hand, transformation improves the performance of the recommender slightly when there are no new ratings added. With 100% new ratings added, transformation yields

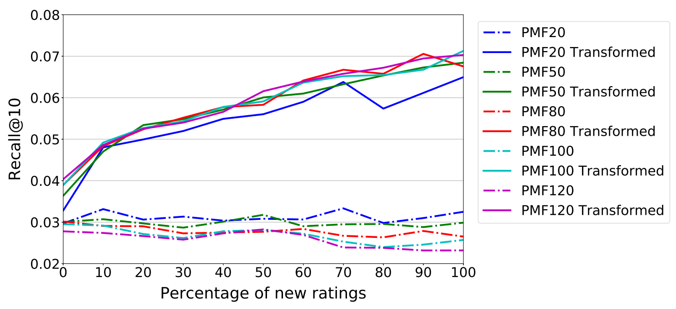


(a) PMF rank 10 (b) PMF rank 50 (c) PMF rank 120

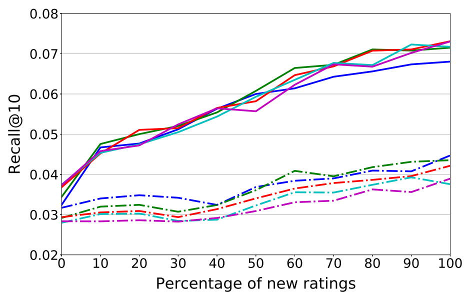
Figure 4.6: PMF latent factors of the attacked rating matrix are transformed into SVD latent factor space.



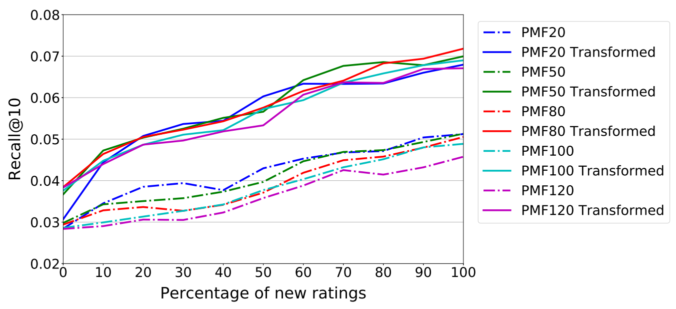
(a) SVD rank 50



(b) SVD rank 100



(c) SVD rank 150



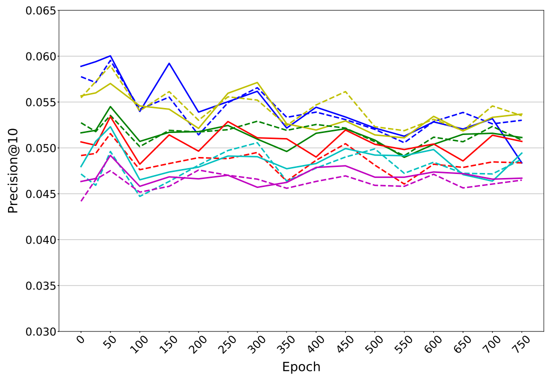
(d) SVD rank 200

Figure 4.7: Transformation of PMF latent factors into SVD latent factor space vs. adding new rating values from SVD low-rank reconstruction

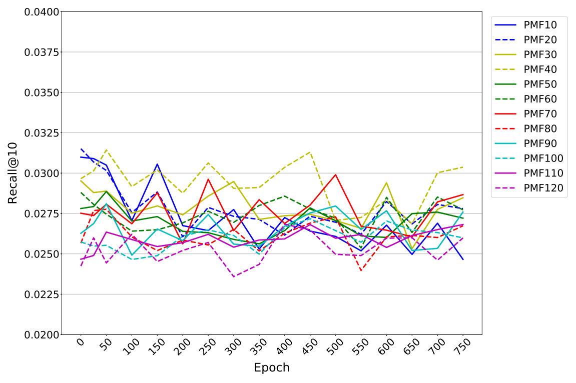
significant improvement in recall@10. Therefore, transformation and adding new ratings do not have a significant effect individually, but together they improve the result significantly.

### Impact of fake profiles learned by GAN

In this section, we explore how fake profiles learned at different epochs of GAN affect the performance of the recommender system. In this experiment we only consider fake profiles generated by GAN and the second step of the attack which is iterative gradient is not considered here. Performance of the recommender system with respect to recall@10 and precision@10 is illustrated in Figure 4.8. The performance of the recommender system does not vary significantly over user profiles learned from various epochs of GAN. Performance corresponding to epoch value 50 is slightly higher.



(a) Precision@10



(b) Recall@10

Figure 4.8: Impact of fake profiles learned at different epoch of GAN on the performance of the recommender system.

## 4.4.3 Combination of Attack I and Attack II

In this section, we are interested to see if our proposed method is able to defend against multiple types of adversarial attacks. Evaluating the performance of our model on a combination of different attacks is vital as in real world recommender systems there could be various attackers with different adversarial objectives who try to inject a group of fake users to serve their malicious purposes. We performed local low-rank reconstruction on the attacked MovieLense 100k dataset with 250 fake users, half generated using attack I (RecSys20) threat model and the other half generated using attack II (RecSys19) model. A summary of results on the combined attacks is reported in Table 4.3. The Combination of the two attacks i.e., RecSys19 + RecSys20, creates a stronger attack that achieves a higher hit ratio on target items compared to the hit ratio of each individual attack. Despite a

| Defense | Attack | Overall HR@10 | Target Items HR@10 |
|---|---|---|---|
| No Defense | None | 0.8293 | 0.1432 |
| | RecSys19 | 0.8271 | 0.1505 |
| | RecSys20 | 0.8346 | 0.1824 |
| | RecSys19 + RecSys20 | 0.8452 | 0.2354 |
| [200, 50] | None | 0.8006 | 0.1697 |
| | RecSys19 | 0.8165 | 0.1241 |
| | RecSys20 | 0.8250 | 0.1654 |
| | RecSys19 + RecSys20 | 0.8293 | 0.1569 |

Table 4.3: Impact of the local SVD reconstruction on the performance of WRMF(SGD) model on MovieLens 100k dataset while various adversarial users are present.

more detrimental attack, our low-rank reconstruction defense is yet able to resist against the attack and reduce the hit ratio on the target items.

# Chapter 5

# Tensor-based Complementary Product Recommendation

Chapter based on material published in IEEE Big Data 2021 [94].

## 5.1 Introduction

Customers face millions of products in an online grocery shopping experience, making the shopping process an exhausting and confusing task for them. Recommender systems are valuable tools that help the customer by narrowing down the search space to products that the customer is desired to see and purchase. Personalized recommender systems are critical components of online shopping platforms. Personalized recommendations based on customers' shopping habits are beneficial to customers and can lead to sales growth. In online grocery shopping, customers often follow repetitive shopping habits. They tend to purchase specific products over and over and rarely switch to other similar products. There-

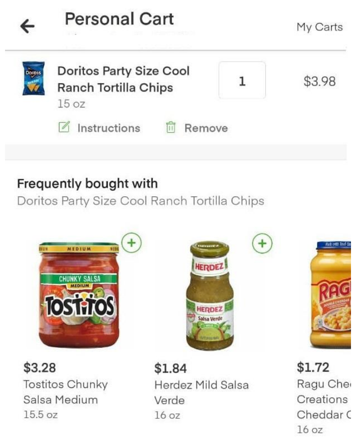


Figure 5.1: Example of complementary products recommended according to the current product in the basket of the customer. Here different types of salsa are recommended to the customer with a bag of chips in his/her shopping cart.

fore, analyzing customers' shopping behavior and providing personalized recommendations is of high importance in an online grocery shopping platform.

One of the main types of personalized recommendations is complementary recommendations. Considering a single shopping session, products that are often purchased together in one basket are considered to be complementary to each other. Complementary products are related to each other in some way and together fulfill customer needs. Chips and salsa, burger and burger buns, peanut butter and jelly are examples of complementary products often purchased together. To improve customer's experience during online shopping, it is vital to recommend relevant products according to what currently exists in the customer's basket.

A good complementary recommender system is essential for various reasons:

- **Shopping efficiency**: It helps the customer to build the shopping basket efficiently and reduces exploration time. They can quickly find the relevant products from the recommendation list, instead of having to search for them. Thus, this can help customers save time. According to Instacart platform, on average, it takes about 45 minutes to build a basket. By providing relevant and personalized recommendations, this time can be reduced and the shopping process will get more convenient for the customer.

- **Novel product recommendation**: Novel products for a customer are those that the customer has never purchased before. Sometimes, customers have no idea about what could be complementary to the products in their basket. Such basket-contextual, personalized recommendations can help customers discover novel products, especially at the end of the shopping process.
- **Business growth**: From a business perspective, novel product purchases can help to increase the basket size of the customer and generates incremental Gross Merchandise Value (GMV). Moreover, customers who have a seamless experience are highly likely to come back for future shopping.

This chapter introduces a tensor-based method to address the complementary recommendation problem in online grocery shopping. Tensor is used to represent products co-purchased by customers and tensor decomposition techniques are used to find product and customer embeddings in low-dimensional space. In the next step, the embeddings are used to score products with respect to the current basket of the customer and products with the highest scores are recommended as complementary to the current basket. Tensor-based recommender systems have shown great success by considering multiple aspects of data and incorporating additional information such as context. Tensor modeling and factorization learns a joint representation of the items and the context, which has been shown to result in richer representations that can provide better estimation for missing ratings/scores [272].

Our contributions are as follows:

1. **Novel tensor-based formulation**: We introduce a tensor-based method that represents complementary product pairs in the form of a three-mode tensor and we use tensor decomposition techniques to infer product and user embeddings.
2. **Efficient Solution**: We consider mini-batch tensors that allow parallel and sequential tensor decomposition to handle large-scale datasets.

## 5.2 Related Work

### 5.2.1 Frequent Purchase Mining

In the field of complementary product recommendation, one basic and trivial method is to recommend products according to their frequency of purchase [127]. In a non-personalized recommendation task, most frequently purchased products by all customers are recommended to the user, whereas in a personalized task, most frequently bought products by the current user are recommended to him/her. In both cases, recommended products ignore the current basket content.

### 5.2.2 Collaborative Filtering and Matrix Factorization

Many of the recent work use collaborative filtering and matrix factorization techniques to model user-product and product-product relationships. Basket-sensitive Factorization Machine (BFM) and constrained BFM (CBFM) methods use a combination of matrix factorization and association rules to provide complementary recommendations [170]. Collaborative filtering and matrix factorization technique only consider user-item interactions and do not take advantage of additional information available such as product/user features and contextual information. However, tensor-based recommender systems are able to incorporate this additional information and improve the performance of the recommendation. When there is an inherent structure between the interactions, then unfolding that structure may not be efficient in terms of our ability to learn a good representation with the given amount of data. This observation has also been shown in non-factorization-based scenarios where structure is not ignored and helps learn better recommenders [46]. Powerful recommender systems also consider contextual features and matrix factorization techniques only consider first-order interaction of users and items and ignore the additional contextual features that can improve personalized recommendation. For instance, considering contextual information such as time and location lead to stronger recommendations [150], while matrix factorization methods cannot be easily adapted to leverage such information.

### 5.2.3 Representation Learning

Another group of studies use popular word representation learning techniques in NLP, like skip-gram, to generate product recommendations. Liang et al. [179] combined matrix factorization and word2vec item embeddings to learn product recommendations. **Item2vec** [42] is an extension of word2vec that infers item-item relations by learning items representations in a low-dimensional space. **Prod2vec** [121] is another method in this category that learns product representations from user purchase histories. An important characteristic of a complementary recommender system is to jointly learn product-product and user-product relations, and the aforementioned methods fail in this aspect . In a three-mode tensor representation where one aspect is product-product relationships and another aspect is user-product interactions, tensor decomposition provides latent factors that capture the hidden structure of data by jointly optimizing on both aspects. For instance, in a three-mode tensor, tensor factorization's objective is as follows:

$$\underset{A,B,C}{minimize} \|\underline{\mathbf{X}} - [\![A, B, C]\!]\|_F^2 \tag{5.1}$$

where $A$, $B$, and $C$ are latent factor matrices derived from the tensor factorization and the factor matrices that minimize the objective function are learned at the same time.

One of the state-of-the-art methods in complementary product recommendation is the **triple2vec** method [268]. This method also utilizes skip-gram embedding learning framework. Triple2vec performs Skip-gram with negative sampling over (product i, product j, user u) triples. Two products co-purchased in a basket by a user form a triple. Triples are used to generate product and user embeddings. Triple2vec inference time increases with basket size. To address this problem, **RTT2vec** (Real-Time Triple2vec) [188] was proposed by Mantha et al. that transforms inference into a similarity search problem and improves the inference time by utilizing approximate nearest neighbor indexing methods such as ANNOY, Faiss, and ScaNN[34], [148], [124].

### 5.2.4 Tensor-based Recommenders

Tensor-based methods can be considered as an extension of matrix factorization recommender system . In matrix factorization, we are dealing with 2-dimensional data, while in tensor factorization techniques, data is represented in higher dimensions ($\geq 3$). Tensor-based methods are able to analyze multiple aspects of data simultaneously and jointly. Matrix factorization models extract user-product interactions, while tensors are able to capture multi-aspect interactions. Tensors are great tools to represent multidimensional data and by considering multiple aspects of data into decomposition, they have been successful in recommender systems [150, 113],[66, 286]. In this chapter, we leverage the multi-aspect property of tensors to model product-product interaction within each basket for different users as a three-mode tensor. Decomposing this tensor allows us to find latent components that reveal product-product and user-product interaction to infer personalized complementary recommendations.

## 5.3 Proposed Method

### 5.3.1 Tensor decomposition to learn product and user embeddings

Let $P = \{p_1, p_2, ..., p_M\}$ be the set of $M$ products and $U = \{u_1, u_2, ..., u_N\}$ be the set of $N$ users. Given $B_u \subset P$, the set of products in the current basket of user $u \in U$, the goal of complementary product recommendation is to recommend top-k products $R = \{p_1^*, p_2^*, ..., p_k^*\}$ such that $p_j^* \notin B_u$. Theses $p_j^*$ products are considered complementary to products in the current basket $B_u$.

To model complementary products, we consider product-to-product relationships for each user . Representing product-to-product relationships per user allows us to provide personalized complementary recommendations. Two products that are always purchased together by a user may not be complementary for another user. For instance, user $A$ mostly purchases chips and salsa, but user $B$ purchases chips and guacamole most of the time. If we only consider global product-to-product relationships, complementary recommendations may be compatible with the need of the majority of users, but it does not fulfill the needs of users who do not behave like others and have their own preferences. For instance, assume most of the users consume meat, but there are small number of users who are vegetarian. By only considering the global product-to-product relations, we ignore the minority group. However, someone who is vegetarian does not want to see meat recommendations. Therefore, we require to capture the behavior of each user separately and recommend products compatible with their shopping profile. We represent this information in the form of a three-mode tensor (three-dimensional array) $\underline{\mathbf{X}}$. In this chapter, A tensor is denoted by an underlined bold uppercase. Next, we learn product and user embeddings using tensor decomposition techniques. The idea behind tensor factorization is to represent users and products in a lower-dimensional space. Each element of the three-mode tensor $\underline{\mathbf{X}}$ represents the number of times two products have been purchased together by a user:

$$\underline{\mathbf{X}}\,(i,j,k) = c_{ijk}; \quad p_i, p_j \in P \text{ and } u_k \in U \tag{5.2}$$

where $c_{ijk}$ is the number of times that user $u_k$ has purchased two products $p_i$ and $p_j$ together.

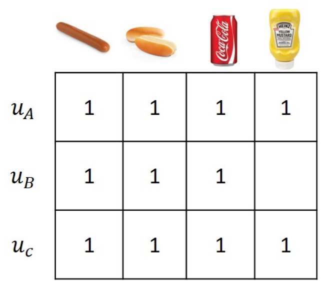
To capture complementary relationships between products, we need to track products that are co-purchased in a single basket. Traditional matrix factorization methods ignore such information and only consider user-product interactions. To better elaborate the difference between matrix and tensor representations, consider the following example. Given 3 users $u_A$, $u_B$, and $u_c$, assume the following transactions for them:

- User $u_A$ performs one transaction:

- Hot dog, hot dog buns, coke, and mustard

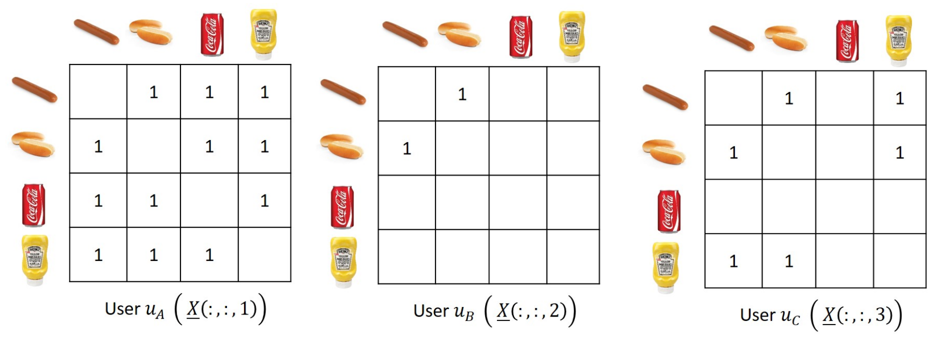
- User $u_B$ performs the following three transactions separately:
  - Basket #1: Hot dog and hot dog buns
  - Basket #2: Coke
  - Basket #3: Mustard
- User $u_C$ performs a single transaction:
  - Hot dog, hot dog buns, and mustard

Figure 5.2 shows matrix vs. tensor representation corresponding to the aforementioned example. Using the matrix representation, users $u_A$ and $u_B$ are exactly similar



| | | | | |
|---|---|---|---|---|
| $u_A$ | 1 | 1 | 1 | 1 |
| $u_B$ | 1 | 1 | 1 | |
| $u_c$ | 1 | 1 | 1 | 1 |

(a) Matrix representation



User $u_A$ $\left(\underline{X}(:,:,1)\right)$

| | | | |
|---|---|---|---|
| | 1 | 1 | 1 |
| 1 | | 1 | 1 |
| 1 | 1 | | 1 |
| 1 | 1 | 1 | |

User $u_B$ $\left(\underline{X}(:,:,2)\right)$

| | | | |
|---|---|---|---|
| | 1 | | |
| 1 | | | |
| | | | |
| | | | |

User $u_C$ $\left(\underline{X}(:,:,3)\right)$

| | | | |
|---|---|---|---|
| | 1 | | 1 |
| 1 | | | 1 |
| | | | |
| 1 | 1 | | |

(b) Tensor representation

Figure 5.2: (a) Matrix representation vs. (b) Tensor representation of transactions data.

because they have purchased the same products. However, using the tensor representation, users $u_A$ is more similar to $u_C$ than user $u_B$, and this is what we expect as users $u_A$ and $u_B$ have performed similar transactions. Classical matrix factorization methods predict the probability of recommending an item given a user ($P(item|user)$, whereas for the task of complementary recommendation, we are interested in computing the probability of recommending an item given a user and their current basket ($P(item|user, basket)$).

Traditional and popular tensor decomposition like CANDECOMP/PARAFAC (CP) [129] and Tucker [265] generate three different latent matrices corresponding to each mode of the tensor. Here, in our problem, the first two modes of tensor $\underline{\mathbf{X}}$ are identical and corresponds to products. Therefore, we only require two of the latent factor matrices. Tucker-2 is a restricted form of Tucker decomposition in which two of the factor matrices are equal. Another decomposition technique that can be applied to our problem is RESCAL [205]. RESCAL has been used to learn the inherent structure of relational data. Here, we are also interested in learning the relationship between products purchased together and RESCAL tensor decomposition method is able to capture this type of relationship between products. Interested readers may refer to [217, 160] for a detailed comparison of tensor decomposition techniques.

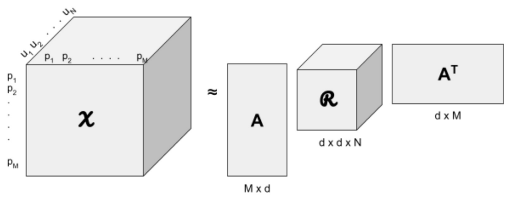


Figure 5.3: RESCAL tensor decomposition

Figure 5.3 shows the RESCAL decomposition of user-product tensor $X$. RESCAL decomposition can be formulated as follows:

$$\underline{\mathbf{X}} \approx A\underline{\mathbf{R}}A^T \tag{5.3}$$

$$\underline{\mathbf{X}}_k = A\underline{\mathbf{R}}_k A^T \tag{5.4}$$

where $A$ is an $M \times d$ latent factor matrix that contains products embeddings and tensor $\underline{\mathbf{R}}$ which is an $d \times d \times N$ is the latent factor corresponding to user embeddings. $\underline{\mathbf{X}}_k = \underline{\mathbf{X}}(:,:,k)$ is called a frontal slice of the tensor $\underline{\mathbf{X}}$ and represents product co-purchased by user $u_k$. Likewise, frontal slice $\underline{\mathbf{R}}_k = \underline{\mathbf{R}}(:,:,k)$ is the user embedding corresponding to user $u_k$.

The factor matrices $A$ and $\underline{\mathbf{R}}$ are computed by solving the regularized minimization problem [205]:

$$min_{A,\underline{\mathbf{R}}_k} f(A, \underline{\mathbf{R}}_k) + g(A, \underline{\mathbf{R}}_k) \tag{5.5}$$

$$f(A, \underline{\mathbf{R}}_k) = \frac{1}{2}\left(\sum_k \|\underline{\mathbf{X}}_k - A\underline{\mathbf{R}}_k A^T\|_F^2\right) \tag{5.6}$$

$$g(A, \underline{\mathbf{R}}_k) = \frac{1}{2}\left(\|A\|_F^2 + \sum_k \|\underline{\mathbf{R}}_k\|_F^2\right) \tag{5.7}$$

where $f(A, \underline{\mathbf{R}}_k)$ tries to minimise the distance and $g(A, \underline{\mathbf{R}}_k)$ is the regularization part to avoid overfitting.

The product embedding matrix $A$ is shared across all users and by taking the dot product of this matrix and its transpose, products most frequently bought together will have a higher score. On the other hand, the user embedding matrix $\underline{\mathbf{R}}_k$ captures the interaction between products that are mostly purchased by a specific user $u_k$. Therefore, the matrix $\underline{\mathbf{R}}_k$ is used to adjusts the product-to-product scores for the user $u_k$ and therefore product $p_j$

that maximizes $A_i\underline{\mathbf{R}}_kA_j^T$ is the personalized complementary product with respect to product $p_i$ . To elaborate the idea further, consider the element-wise form of the equation 5.6:

$$f(A,\underline{\mathbf{R}}_k) = \frac{1}{2}\left(\sum_{i,j,k}\left(\underline{\mathbf{X}}_{ijk} - A_i\underline{\mathbf{R}}_kA_j^T\right)^2\right) \tag{5.8}$$

where $A_i = A(i,:)$ and $A_j = A(j,:)$ are rows of latent factor matrix $A$ that are embedding vectors of length $d$ corresponding to products $p_i$ and $p_j$, respectively. Assume, user $u_k$ currently has product $p_i$ in their basket and our goal is to find a product that is complementary to $p_i$. To minimize $f(A,R_k)$, the term $A_i\underline{\mathbf{R}}_kA_j^T$ should be as close as possible to the value $\underline{\mathbf{X}}_{ijk}$. If products $p_i$ and $p_j$ are frequently purchased together by user $u_k$, the value of $\underline{\mathbf{X}}_{ijk}$ will be large and therefore the term $A_i\underline{\mathbf{R}}_kA_j^T$ should be maximized. Given a product $p_i$, its complementary product $p_j$ will have an embedding which is closest to the embedding corresponding to $p_i$, i.e. the dot product of $A_iA_j^T$ will have the highest score. Thus, product(s) that maximize the following equations are considered as top $N$ complementary products with respect to product $p_i$:

$$\underset{j\in P\setminus B_{u_k}}{\arg\max} A_i\underline{\mathbf{R}}_kA_j^T \qquad i \in B_{u_k} \tag{5.9}$$

### 5.3.2 Optimizing RESCAL Decomposition

The algorithm to compute factor matrices in RESCAL decomposition performs alternating updates of matrices $A$ and $\underline{\mathbf{R}}_k$ for all $k$ until $\frac{f(A,\underline{\mathbf{R}}_k)}{\|X\|_F^2}$ converges to some small threshold or a maximum number of iterations is exceeded.

For a large-scale dataset with millions of users and thousands of products, we will have a huge sparse tensor, and the alternating algorithm is very slow and inefficient. RESCAL is a restricted form of TUCKER decomposition in which one of the modes is left uncompressed, i.e., one of the latent factors is the identity matrix. This restricted Tucker decomposition is known as Tucker-2 [265]. To speed up the decomposition algorithm, we use Higher-Order Singular Value Decomposition (HOSVD) algorithm [141]to approximate factor matrices. HOSVD algorithm does not compute the optimal solution, however, it is very popular due to its simplicity. HOSVD algorithm computes the factor matrices by performing singular value decomposition on the matricized form of the tensor across each mode (dimension).

Moreover, performing tensor decomposition on such a large tensor requires lots of memory. To solve this problem, instead of performing decomposition on a single tensor containing all users' data, we split the tensor into smaller batches that only contain a subset of users and perform decomposition on each batch separately. This allows us to run the decomposition on datasets with millions of users. Also, adding new users to the dataset does not require retraining the model on the entire dataset and we can only train our model on the batch of recent users.

### 5.3.3 Optimizing Inference Time

To achieve real-time inference, we need further improvements. With a large number of products in the dataset, computing product scores is very time-consuming and model inference time increases with the basket size. To find top-k products that maximize the score with respect to the current basket, we need to perform dot product between basket product embeddings and all other products in the dataset. To speed up the process, we use hashing technique by using Approximation Nearest Neighbor (ANN) indexing library, ANNOY[1]. This allows us to perform the approximate dot product efficiently. The dot product of two vectors is maximized when they are most similar to each other. We create the ANN index on all products in the dataset. For each basket, the query vector $Q_i$ is $A_i \underline{\mathbf{R}}_k$for all products $p_i \in B_{u_k}$ and we would like to find the products in the dataset which are closest to the query vector. Therefore, the inference problem can be rewritten as follows:

$$\begin{aligned} &\underset{j \in ANN(Q_i)}{\arg\max}\ Q_i A_j^T \\ &Q_i = A_i \underline{\mathbf{R}}_k; \qquad i \in B_{u_k} \end{aligned} \tag{5.10}$$

Now, instead of searching through all products in the dataset to find top-K recommendations, we only need to search through $L$ products where $L = |ANN(Q_i)| << M$.

## 5.4 Experiments

### 5.4.1 Dataset and Experiment Setup

In our experiments we used Instacart public dataset published for Kaggle competition in 2017[2]. Th statistics of the Instacart dataset is reported in Table 5.1.

To split transaction data into train/validation/test sets, we follow the setting mentioned in [268]: Transactions are sorted chronologically. For each user, the most recent transaction is used for testing, the second-to-last transaction is used for validation and all

[1] https://github.com/spotify/ annoy

[2] https://www.kaggle.com/c/instacart-market-basket-analysis/data

|  |  |
|---|---|
| **No. of transactions** | 3,345,786 |
| **No. of users** | 206,209 |
| **No. of products** | 49,684 |
| **No. of products purchased at least 10 times** | 42,987 |
| **Average basket size** | 10.10 |

Table 5.1: Instacart Dataset statistics

other transactions are used for training. For users with only one transaction, it will be used for training, and for users with two transactions, the first transaction is used for training and the last one is used for testing.

### 5.4.2 Evaluation

#### Metrics

We evaluate the performance of models with the following metrics: Recall@K, NDCG@K, and Precision@K. Recall@K measures the fraction of relevant items correctly recommended in the top-K items. NDCG@K (Normalized Discounted Cumulative Gain) is a ranking metric that uses position in the recommendation list to measure gain. Metrics are reported at K=10.

#### Results

To evaluate the performance of our method on the test dataset, we randomly select 80% of the products in the basket as input, and the rest of the products are used as reference for evaluation.

| Method | Recall@10 | NDCG@10 | Precision@10 |
|---|---|---|---|
| Popularity | 0.104 | 0.081 | 0.029 |
| triple2vec (d = 32) | 0.103 | 0.162 | 0.022 |
| tensor (d = 32) | 0.145 | 0.172 | 0.037 |
| triple2vec (d =128) | 0.149 | 0.178 | 0.030 |
| tensor (d=128) | **0.192** | **0.193** | **0.047** |

Table 5.2: Peformance of the proposed model vs. popularity and triple2vec baseline methods.

We compare our method with two baselines: **popularity**, and **triple2vec** [268]. Popularity method always recommends the most frequently purchased products by all users. For tensor-based and triple2vec methods, we report results for two different embedding dimensions 32 and 128 to better understand the impact of embedding dimension on the recommendation performance. Table 5.2 shows the results. Popularity does not consider the context of the current basket and always recommends the most frequent products which might be irrelevant with respect to the current basket. The popularity method memorizes frequent purchases and lacks generalization power since it is unable to capture product semantics. However, customers are often loyal to certain products and brands and repeatedly purchase the same products and are reluctant to switch to some other products. Therefore, the popularity method achieves an acceptable performance by memorizing the shopping history of users. Our proposed tensor-based method outperforms popularity and triple2vec in all cases. A larger value of the embedding dimension (d=128 vs. d=32) helps to better estimate the original tensor and thus improves the performance of the model.

We also evaluate the proposed method and triple2vec for novel product recommendations. Novel purchases are products that customer is purchasing for the first time. Introducing novel products to customers and encouraging them to purchase more products leads to larger basket size and higher GMV for the business. In the novel product recommendation experiment, we are interested to see how many of the top K recommended items are novel and the customer has never purchased them before. Sometimes customers are not aware of possible complementary products and novel recommendations give them the chance to try new complementary products that are frequently purchased by other customers. For the test data in this experiment, we only considered transactions including at least 3 novel products. These 3 novel products are used for inference and the rest of the basket is used as input of the recommender system. The result for the novel complementary recommendation is shown in Table 5.3. Both tensor-based method and triple2vec perform poorly in recommending novel products. They mostly recommend products that have already been purchased by the user. For future work, we aim to utilized coupled matrix-tensor factorization to leverage product features and improve the performance of novel product recommendations.

| Method | Recall@10 | NDCG@10 | Precision@10 |
|---|---|---|---|
| triple2vec-novel (d = 32) | 0.012 | 0.0 | 0.005 |
| triple2vec-novel (d = 128) | 0.015 | 0.0 | 0.007 |
| tensor-novel (d = 32) | 0.010 | 0.0 | 0.005 |
| tensor-novel (d = 128) | 0.013 | 0.0 | 0.006 |

Table 5.3: Performance of the proposed model vs. triple2vec on novel complementary product recommendation.

### Handling Cold-Start Users

For new customers, we do not have a shopping history; therefore, it is not possible to have a personalized recommendation. In this case, we can either have a non-personalized recommendation and score products just based on the product embedding, or consider the average of all users' embeddings as the user embedding for the new customer. In Table 5.4 we report the performance of the proposed model for the two mentioned cases. Here we only report the result for the embedding dimension 128 that achieves a higher performance.

| Method | Recall@10 | NDCG@10 | Precision@10 |
|---|---|---|---|
| tensor (d = 128)-average user | 0.122 | 0.271 | 0.033 |
| tensor (d = 128)-non-personalized | 0.120 | 0.266 | 0.032 |

Table 5.4: Performance of tensor-based method in handling cold-start users.

The performance of personalized and non-personalized approaches to handle new users are about the same and both have an acceptable performance to deal with cold-start users.

**Evaluating inference time**

Here, We compare the inference time of the exact approach with the approximate method using ANNOY library. Table 5.5 compares these two approaches for different basket sizes. The exact inference method which computes dot products between all product embedding, gets very slow as the basket size increases and makes it inapplicable.

|  | Inference Time (sec) |  |
|---|---|---|
| Basket size | Exact method | ANNOY |
| 10 | 1.089 | 0.013 |
| 20 | 2.165 | 0.027 |
| 30 | 3.307 | 0.053 |
| 40 | 4.344 | 0.052 |
| 50 | 5.345 | 0.083 |
| 60 | 6.454 | 0.065 |
| 70 | 7.557 | 0.077 |
| 80 | 8.527 | 0.093 |
| 90 | 9.725 | 0.086 |
| 100 | 11.387 | 0.088 |
| 110 | 12.513 | 0.138 |
| 120 | 13.069 | 0.164 |

Table 5.5: Exact inference vs. approximate inference using ANNOY.

# Chapter 6

# Conclusions

## 6.1 Conclusions

In this dissertation, we attempted to examine the characteristics of adversarial attacks in various domains, including graphs, images, and recommender systems and we aimed to find a unifying theme to defend against the adversarial attacks in these domains.

In Chapter 2, we investigated the properties of NETTACK perturbations for graphs. Due to the vulnerability of the node classification approaches to the adversarial attacks, we highlighted the need for a defense system or robust node classification methods. We illustrated that NETTACK generates high-rank perturbations that can be discarded using a low-rank approximation of the adjacency and feature matrices. We showed that a rank-10 approximation of the matrices is able to defend adversarial attacks with a high probability and achieve a performance close to the performance on the clean graph. Furthermore, we examined the robustness of t-PINE, a tensor-based node embedding against NETTACK and we observed that it is very robust for lower embedding dimensions and the robustness of the embedding decreases as the dimension gets bigger. In addition, we proposed an algorithm to generate low-rank adversarial attacks that could fool both GCN and tensor-based embeddings. *LowBlow* perturbations are noticeable and the attacked graph does not have the same degree distribution as the input graph. We also showed that modifying *LowBlow* to only keep the edges that preserve the degree distribution makes it a high-rank attack similar to NETTACK. In conclusion, unnoticeable adversarial attacks on graphs impose high-rank changes in singular values of the input graph which can be greatly eliminated with our proposed low-rank defense mechanism.

In Chapter 3, we explored adversarial attacks on images and examined to what extent low-rank tensor decomposition of perturbed images during the preprocessing step helps to defend against adversarial attacks. The low-rank approximation of the perturbed image is then fed to the deep network for the task of classification. We evaluated our method against popular adversarial attacks: FGSM, I-FGSM, and PGD. We illustrated that considering images in small batches better captures the latent structure of perturbations and helps to improve the performance of the model. We also showed that how different configurations allow to trade-off between accuracy and runtime.

In Chapter 4 of this dissertation, we attempted to defend against adversarial attacks on recommender systems. we investigated two adversarial attacks on recommender systems and demonstrated that low-rank reconstruction and transforming into SVD space helps to reduce the impact of the attack and improve the performance of the recommender system. We also demonstrated that our low-rank solution can resist adversarial users generated by different threat models.

Chapter 5, evaluates the effectiveness of low-rank solutions on recommender systems to address the problem of complementary product recommendation. To this aim, we proposed a tensor-based model to address the complementary product recommendation task in online grocery shopping. A three-mode tensor was used to model product-product co-purchases by users. Next, RESCAL tensor decomposition technique was used to learn latent factors of the tensor that correspond to product and user embeddings. These embeddings were utilized to infer complementary products given a current basket of a user. To improve the training process to handle large-scale datasets, we performed decomposition on small tensors including a subset of users and their transactions. Decomposition of mini-batches allows to train the model both in parallel and sequentially. If enough memory is available, one can take advantage and run the decomposition in parallel. However, in limited memory systems, we can perfectly run the model sequential but in a longer time. Moreover, we leveraged the approximate nearest neighbor indexing library, ANNOY, to speed up the inference process and allow real-time inference.

# Bibliography

[1] Aminer datasets for social network analysis. https://aminer.org/data-sna.

[2] Apache hadoop. http://hadoop.apache.org/.

[3] Apache reef. http://reef.apache.org/.

[4] Apache spark. http://spark.apache.org/.

[5] Apache storm. http://storm.apache.org/.

[6] Dagstuhl perspectives workshop: Tensor computing for internet of things. http://www.dagstuhl.de/en/program/calendar/semhp/?semnr=16152.

[7] Factoring tensors in the cloud: A tutorial on big tensor data analytics. http://www.cs.cmu.edu/~epapalex/tutorials/icassp14.html.

[8] Google tensorflow. https://www.tensorflow.org/.

[9] T-Drive trajectory data sample. http://research.microsoft.com/apps/pubs/?id=152883.

[10] Tricap: ThRee-way methods in chemistry and psychology 7th edition, 2015. http://people.ece.umn.edu/~nikos/TRICAP_home.html. Accessed: May 21, 2016.

[11] WeChat. http://web.wechat.com.

[12] Workshop on tensor decompositions and applications. http://www.esat.kuleuven.be/stadius/TDA2016/. Accessed: May 21, 2016.

[13] Enron e-mail dataset. http://www.cs.cmu.edu/~enron/, Last accessed: 9/9/2014.

[14] Read the web. http://rtw.ml.cmu.edu/rtw/, Last accessed: 9/9/2014.

[15] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.

[16] E. Acar, C. Aykut-Bingol, H. Bingol, R. Bro, and B. Yener. Multiway analysis of epilepsy tensors. *Bioinformatics*, 23(13):i10–i18, 2007.

[17] Evrim Acar, Seyit A. Çamtepe, Mukkai S. Krishnamoorthy, and Bülent Yener. Modeling and multiway analysis of chatroom tensors. In *Intelligence and Security Informatics*, pages 256–268. Springer, 2005.

[18] Evrim Acar, Daniel M Dunlavy, and Tamara G Kolda. A scalable optimization approach for fitting canonical tensor decompositions. *Journal of Chemometrics*, 25(2):67–86, 2011.

[19] Evrim Acar, Daniel M Dunlavy, Tamara G Kolda, and Morten Mørup. Scalable tensor factorizations with missing data. In *SDM*, pages 701–712. SIAM, 2010.

[20] Evrim Acar, Tamara G Kolda, and Daniel M Dunlavy. All-at-once optimization for coupled matrix and tensor factorizations. *arXiv preprint arXiv:1105.3422*, 2011.

[21] Evrim Acar, Anders J Lawaetz, Morten Rasmussen, and Rasmus Bro. Structure-revealing data fusion model with applications in metabolomics. In *Engineering in Medicine and Biology Society (EMBC), 2013 35th Annual International Conference of the IEEE*, pages 6023–6026. IEEE, 2013.

[22] Evrim Acar, Evangelos E Papalexakis, Morten A Rasmussen, Anders J Lawaetz, Mathias Nilsson, and Rasmus Bro. Structure-revealing data fusion. *BMC bioinformatics*, 15(1):239, 2014.

[23] Lada A Adamic and Natalie Glance. The political blogosphere and the 2004 us election: divided they blog. In *Proceedings of the 3rd international workshop on Link discovery*, pages 36–43. ACM, 2005.

[24] Rakesh Agrawal, Behzad Golshan, and Evangelos E. Papalexakis. A study of distinctiveness in web results of two search engines. In *WWW'15 Web Science Track*, (*author order is alphabetical*).

[25] Rakesh Agrawal, Behzad Golshan, and Evangelos E. Papalexakis. Whither social networks for web search? In *ACM KDD'15*, (*author order is alphabetical*).

[26] Rakesh Agrawal, Behzad Golshan, and Evangelos Papalexakis. A study of distinctiveness in web results of two search engines. Technical Report TR-2015-001, Data Insights Laboratories, San Jose, California, January 2015.

[27] Rakesh Agrawal, Behzad Golshan, and Evangelos Papalexakis. Overlap in the web search results of google and bing. *The Journal of Web Science*, 2(2):17–30, 2016.

[28] Naveed Akhtar and Ajmal Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, 6:14410–14430, 2018.

[29] L. Akoglu, M. McGlohon, and C. Faloutsos. Oddball: Spotting anomalies in weighted graphs. *Advances in Knowledge Discovery and Data Mining*, pages 410–421, 2010.

[30] Saba A Al-Sayouri, Ekta Gujral, Danai Koutra, Evangelos E Papalexakis, and Sarah S Lam. t-pne: Tensor-based predictable node embeddings. In *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 491–494. IEEE, 2018.

[31] Animashree Anandkumar, Rong Ge, Daniel Hsu, Sham M Kakade, and Matus Telgarsky. Tensor decompositions for learning latent variable models. *The Journal of Machine Learning Research*, 15(1):2773–2832, 2014.

[32] C.A. Andersson and R. Bro. The n-way toolbox for matlab. *Chemometrics and Intelligent Laboratory Systems*, 52(1):1–4, 2000.

[33] Miguel Araujo, Spiros Papadimitriou, Stephan Günnemann, Christos Faloutsos, Prithwish Basu, Ananthram Swami, Evangelos E. Papalexakis, and Danai Koutra. Com2: Fast automatic discovery of temporal ('comet') communities. In *Advances in Knowledge Discovery and Data Mining*. Springer, 2014.

[34] Martin Aumüller, Erik Bernhardsson, and Alexander Faithfull. Ann-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. In *International Conference on Similarity Search and Applications*, pages 34–49. Springer, 2017.

[35] Brett W Bader, Richard A Harshman, and Tamara G Kolda. Temporal analysis of semantic graphs using asalsan. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 33–42. IEEE, 2007.

[36] Brett W. Bader and Tamara G. Kolda. Efficient MATLAB computations with sparse and factored tensors. *SIAM Journal on Scientific Computing*, 30(1):205–231, December 2007.

[37] B.W. Bader, M.W. Berry, and M. Browne. Discussion tracking in enron email using parafac. *Survey of Text Mining II*, pages 147–163, 2008.

[38] B.W. Bader, R.A. Harshman, and T.G. Kolda. Temporal analysis of social networks using three-way dedicom. *Sandia National Laboratories TR SAND2006-2161*, 2006.

[39] B.W. Bader and T.G. Kolda. Matlab tensor toolbox version 2.2. *Albuquerque, NM, USA: Sandia National Laboratories*, 2007.

[40] Pierre Baldi. Autoencoders, unsupervised learning, and deep architectures. In *Proceedings of ICML workshop on unsupervised and transfer learning*, pages 37–49, 2012.

[41] Grey Ballard, Tamara G. Kolda, and Todd Plantenga. Efficiently computing tensor eigenvalues on a GPU. In *IPDPSW'11: Proceedings of the 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and PhD Forum*, pages 1340–1348. IEEE Computer Society, May 2011.

[42] Oren Barkan and Noam Koenigstein. Item2vec: neural item embedding for collaborative filtering. In *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2016.

[43] M. Barnathan, V. Megalooikonomou, C. Faloutsos, S. Faro, and F.B. Mohamed. Twave: High-order analysis of functional mri. *NeuroImage*, 2011.

[44] R. Bekkerman, M. Bilenko, and J. Langford. *Scaling Up Machine Learning: Parallel and Distributed Approaches*. Cambridge University Press, 2011.

[45] Alessandro Bessi. Two samples test for discrete power-law distributions. *arXiv preprint arXiv:1503.00643*, 2015.

[46] Alex Beutel, Paul Covington, Sagar Jain, Can Xu, Jia Li, Vince Gatto, and Ed H Chi. Latent cross: Making use of context in recurrent recommender systems. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 46–54, 2018.

[47] Alex Beutel, Abhimanu Kumar, Evangelos E. Papalexakis, Partha Pratim Talukdar, Christos Faloutsos, and Eric P Xing. Flexifact: Scalable flexible factorization of coupled tensors on hadoop. In *SIAM SDM'14*, 2014.

[48] Smriti Bhagat, Graham Cormode, and S Muthukrishnan. Node classification in social networks. In *Social network data analytics*, pages 115–148. Springer, 2011.

[49] Arjun Nitin Bhagoji, Daniel Cullina, and Prateek Mittal. Dimensionality reduction as a defense against evasion attacks on machine learning classifiers. *arXiv preprint*, 2017.

[50] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389*, 2012.

[51] Aleksandar Bojcheski and Stephan Günnemann. Adversarial attacks on node embeddings. *arXiv preprint arXiv:1809.01093*, 2018.

[52] Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. 2018.

[53] R. Bro. Parafac. tutorial and applications. *Chemometrics and intelligent laboratory systems*, 38(2):149–171, 1997.

[54] R Bro, ND Sidiropoulos, and GB Giannakis. A fast least squares algorithm for separating trilinear mixtures. In *Int. Workshop Independent Component and Blind Signal Separation Anal*, pages 11–15, 1999.

[55] Rasmus Bro. *Multi-way analysis in the food industry: models, algorithms, and applications*. PhD thesis, 1998.

[56] Rasmus Bro and Henk AL Kiers. A new efficient method for determining the number of components in parafac models. *Journal of chemometrics*, 17(5):274–286, 2003.

[57] Paul E Buis and Wayne R Dyksen. Efficient vector and parallel manipulation of tensor products. *ACM Transactions on Mathematical Software (TOMS)*, 22(1):18–23, 1996.

[58] Rajmonda Sulo Caceres and Tanya Berger-Wolf. Temporal scale of dynamic networks. In *Temporal Networks*, pages 65–94. Springer, 2013.

[59] Rajmonda Sulo Caceres, Tanya Berger-Wolf, and Robert Grossman. Temporal scale of processes in dynamic networks. In *2011 IEEE 11th International Conference on Data Mining Workshops*, pages 925–932. IEEE, 2011.

[60] Deng Cai, Xiaofei He, and Jiawei Han. Tensor space model for document analysis. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 625–626. ACM, 2006.

[61] J Douglas Carroll and Jih-Jie Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-youn" decomposition. *Psychometrika*, 35(3):283–319, 1970.

[62] Kai-Wei Chang, Wen-tau Yih, and Christopher Meek. Multi-relational latent semantic analysis. In *EMNLP*, pages 1602–1612, 2013.

[63] Kai-Wei Chang, Wen-tau Yih, Bishan Yang, and Christopher Meek. Typed tensor decomposition of knowledge bases for relation extraction. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1568–1579, 2014.

[64] Ian Charest, Rogier A Kievit, Taylor W Schmitz, Diana Deca, and Nikolaus Kriegeskorte. Unique semantic space in the brain of each beholder predicts perceived similarity. *Proceedings of the National Academy of Sciences*, 111(40):14565–14570, 2014.

[65] Ian Charest and Nikolaus Kriegeskorte. The brain of the beholder: honouring individual representational idiosyncrasies. *Language, Cognition and Neuroscience*, 30(4):367–379, 2015.

[66] Huiyuan Chen and Jing Li. Adversarial tensor factorization for context-aware recommendation. In *Proceedings of the 13th ACM Conference on Recommender Systems*, pages 363–367, 2019.

[67] P.A. Chew, B.W. Bader, T.G. Kolda, and A. Abdelali. Cross-language information retrieval using parafac2. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 143–152. ACM, 2007.

[68] Eric C Chi and Tamara G Kolda. On tensors, sparsity, and nonnegative factorizations. *SIAM Journal on Matrix Analysis and Applications*, 33(4):1272–1299, 2012.

[69] Eunjoon Cho, Seth A Myers, and Jure Leskovec. Friendship and mobility: user movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1082–1090, 2011.

[70] Joon Hee Choi and S. Vishwanathan. Dfacto: Distributed factorization of tensors. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 1296–1304. Curran Associates, Inc., 2014.

[71] Konstantina Christakopoulou and Arindam Banerjee. Adversarial attacks on an oblivious recommender. In *Proceedings of the 13th ACM Conference on Recommender Systems*, pages 322–330, 2019.

[72] Aaron Clauset, Cosma Rohilla Shalizi, and Mark EJ Newman. Power-law distributions in empirical data. *SIAM review*, 51(4):661–703, 2009.

[73] Cory Cornelius. The efficacy of shield under different threat models. *arXiv preprint arXiv:1902.00541*, 2019.

[74] Joao Paulo CL da Costa, Martin Haardt, and F Romer. Robust methods based on the hosvd for estimating the model order in parafac models. In *Sensor Array and Multichannel Signal Processing Workshop, 2008. SAM 2008. 5th*, pages 510–514. IEEE, 2008.

[75] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. Adversarial attack on graph structured data. *arXiv preprint arXiv:1806.02371*, 2018.

[76] Nilesh Dalvi, Pedro Domingos, Sumit Sanghai, Deepak Verma, et al. Adversarial classification. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 99–108. ACM, 2004.

[77] Nilaksh Das, Madhuri Shanbhogue, Shang-Tse Chen, Fred Hohman, Siwei Li, Li Chen, Michael E Kounavis, and Duen Horng Chau. Shield: Fast, practical defense and vaccination for deep learning using jpeg compression. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 196–204, 2018.

[78] Ian Davidson, Sean Gilpin, Owen Carmichael, and Peter Walker. Network discovery via constrained tensor analysis of fmri data. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 194–202. ACM, 2013.

[79] André LF De Almeida and Alain Y Kibangou. Distributed large-scale tensor decomposition. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, 2014.

[80] Lieven De Lathauwer and Dimitri Nion. Decompositions of a higher-order tensor in block terms-part iii: Alternating least squares algorithms. *SIAM journal on Matrix Analysis and Applications*, 30(3):1067–1083, 2008.

[81] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. *OSDI*, 2004.

[82] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, September 1990.

[83] Yashar Deldjoo, Tommaso Di Noia, and Felice Antonio Merra. A survey on adversarial recommender systems: from attack/defense strategies to generative adversarial networks. *ACM Computing Surveys (CSUR)*, 54(2):1–38, 2021.

[84] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[85] Chris Ding, Tao Li, and Shenghuo Zhu. Data mining using matrices and tensors (dmmt'08).

[86] Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[87] P. Drineas, R. Kannan, and M.W. Mahoney. Fast monte carlo algorithms for matrices iii: Computing a compressed approximate matrix decomposition. *SIAM Journal on Computing*, 36(1):184, 2006.

[88] Daniel M Dunlavy, Tamara G Kolda, and Evrim Acar. Temporal link prediction using matrix and tensor factorizations. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 5(2):10, 2011.

[89] Gintare Karolina Dziugaite, Zoubin Ghahramani, and Daniel M Roy. A study of the effect of jpg compression on adversarial images. *arXiv preprint arXiv:1608.00853*, 2016.

[90] Nathan Eagle, Alex Sandy Pentland, and David Lazer. Inferring friendship network structure by using mobile phone data. *Proceedings of the National Academy of Sciences*, 106(36):15274–15278, 2009.

[91] C. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[92] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[93] Negin Entezari, Saba A Al-Sayouri, Amirali Darvishzadeh, and Evangelos E Papalexakis. All you need is low (rank) defending against adversarial attacks on graphs. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 169–177, 2020.

[94] Negin Entezari, Evangelos Papalexakis, Haixun Wang, Sharath Rao, and Shishir Kumar Prasad. Tensor-based complementary product recommendation. In *2021 International Conference on IEEE Big Data (Big Data)*. IEEE, 2021.

[95] Negin Entezari and Evangelos E Papalexakis. Tensorshield: Tensor-based defense against adversarial attacks on images. *arXiv preprint arXiv:2002.10252*, 2020.

[96] Dóra Erdos and Pauli Miettinen. Discovering facts with boolean tensor tucker decomposition. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 1569–1572. ACM, 2013.

[97] Dóra Erdos and Pauli Miettinen. Walk'n'merge: A scalable algorithm for boolean tensor factorization. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*, pages 1037–1042. IEEE, 2013.

[98] Evangelos E. Papalexakis, Leman Akoglu, and Dino Ienco. Do more views of a graph help? community detection and clustering in multi-graphs. In *IEEE FUSION'13*.

[99] Evangelos E. Papalexakis, Alex Beutel, and Peter Steenkiste. Network anomaly detection using co-clustering. In *Encyclopedia of Social Network Analysis and Mining*. Springer, 2014.

[100] Evangelos E. Papalexakis and A. Seza Doğruöz. Understanding multilingual social networks in online immigrant communities. WWW '15 Companion.

[101] Evangelos E. Papalexakis and C. Faloutsos. Fast efficient and scalable core consistency diagnostic for the parafac decomposition for big sparse tensors. In *IEEE ICASSP'15*.

[102] Evangelos E. Papalexakis, Christos Faloutsos, and Nicholas D Sidiropoulos. Parcube: Sparse parallelizable tensor decompositions. In *ECML-PKDD'12*.

[103] Evangelos E. Papalexakis., Alona Fyshe, Nicholas D. Sidiropoulos, Partha Pratim Talukdar, Tom M. Mitchell, and Christos Faloutsos. Good-enough brain model: Challenges, algorithms and discoveries in multi-subject experiments. In *ACM KDD'14*.

[104] Evangelos E. Papalexakis, Tom M Mitchell, Nicholas D Sidiropoulos, Christos Faloutsos, Partha Pratim Talukdar, and Brian Murphy. Turbo-smt: Accelerating coupled sparse matrix-tensor factorizations by 200x. In *SIAM SDM'14*.

[105] Evangelos E. Papalexakis, Konstantinos Pelechrinis, and Christos Faloutsos. Location based social network analysis using tensors and signal processing tools. In *IEEE CAMSAP'15*.

[106] Evangelos E. Papalexakis, Nicholas D Sidiropoulos, and Rasmus Bro. From k-means to higher-way co-clustering: multilinear decomposition with sparse latent factors. *IEEE Transactions on Signal Processing*, 2013.

[107] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. In *ACM SIGCOMM Computer Communication Review*, volume 29, pages 251–262. ACM, 1999.

[108] Hadi Fanaee-T and João Gama. Multi-aspect-streaming tensor analysis. *Knowledge-Based Systems*, 89:332–345, 2015.

[109] Minghong Fang, Guolei Yang, Neil Zhenqiang Gong, and Jia Liu. Poisoning attacks to graph-based recommender systems. In *Proceedings of the 34th Annual Computer Security Applications Conference*, pages 381–392, 2018.

[110] Donald W Fausett and Charles T Fulton. Large least squares problems involving kronecker products. *SIAM Journal on Matrix Analysis and Applications*, 15(1):219–227, 1994.

[111] Cédric Févotte and Jérôme Idier. Algorithms for nonnegative matrix factorization with the $\beta$-divergence. *Neural Computation*, 23(9):2421–2456, 2011.

[112] Matt Gardner, Kejun Huang, Evangelos E. Papalexakis., Xiao Fu, Partha Talukdar, Christos Faloutsos, Nicholas Sidiropoulos, and Tom Mitchell. Translation invariant word embeddings. In *EMNLP'15*.

[113] Hancheng Ge, James Caverlee, and Haokai Lu. Taper: A contextual tensor-based approach for personalized expert recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 261–268, 2016.

[114] R. Gemulla, E. Nijkamp, P.J. Haas, and Y. Sismanis. Large-scale matrix factorization with distributed stochastic gradient descent. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 69–77. ACM, 2011.

[115] G.B. Giannakis, P. Stoica, and Y. Hua. *Signal Processing Advances in Wireless and Mobile Communications, Volume 2: Trends in Single-and Multi-User Systems*. Prentice Hall PTR, 2000.

[116] C Lee Giles, Kurt D Bollacker, and Steve Lawrence. Citeseer: An automatic citation indexing system. In *Proceedings of the third ACM conference on Digital libraries*, pages 89–98. ACM, 1998.

[117] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

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

[119] Cyril Goutte and Massih-Reza Amini. Probabilistic tensor factorization and model selection. *Tensors, Kernels, and Machine Learning (TKLM 2010)*, pages 1–4, 2010.

[120] Lars Grasedyck. Hierarchical singular value decomposition of tensors. *SIAM Journal on Matrix Analysis and Applications*, 31(4):2029–2054, 2010.

[121] Mihajlo Grbovic, Vladan Radosavljevic, Nemanja Djuric, Narayan Bhamidipati, Jaikit Savla, Varun Bhagwan, and Doug Sharp. E-commerce in your inbox: Product recommendations at scale. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1809–1818, 2015.

[122] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 855–864. ACM, 2016.

[123] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens Van Der Maaten. Countering adversarial images using input transformations. *arXiv preprint arXiv:1711.00117*, 2017.

[124] Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. Accelerating large-scale inference with anisotropic vector quantization. In *International Conference on Machine Learning*, pages 3887–3896. PMLR, 2020.

[125] Wolfgang Hackbusch and Stefan Kühn. A new scheme for the tensor representation. *Journal of Fourier Analysis and Applications*, 15(5):706–722, 2009.

[126] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017.

[127] Jiawei Han, Hong Cheng, Dong Xin, and Xifeng Yan. Frequent pattern mining: current status and future directions. *Data mining and knowledge discovery*, 15(1):55–86, 2007.

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

[129] R.A. Harshman. Foundations of the parafac procedure: Models and conditions for an" explanatory" multimodal factor analysis. 1970.

[130] Johan Håstad. Tensor rank is np-complete. *Journal of Algorithms*, 11(4):644–654, 1990.

[131] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[132] Lifang He, Xiangnan Kong, S Yu Philip, Ann B Ragin, Zhifeng Hao, and Xiaowei Yang. Dusk: A dual structure-preserving kernel for supervised tensor learning with applications to neuroimages. *matrix*, 3(1):2, 2014.

[133] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182, 2017.

[134] Willem J Heiser. Convergent computation by iterative majorization: theory and applications in multidimensional data analysis. *Recent advances in descriptive multivariate analysis*, pages 157–189, 1995.

[135] Christopher J Hillar and Lek-Heng Lim. Most tensor problems are np-hard. *Journal of the ACM (JACM)*, 60(6):45, 2013.

[136] Joyce C Ho, Joydeep Ghosh, and Jimeng Sun. Marble: high-throughput phenotyping from electronic health records via sparse nonnegative tensor factorization. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 115–124. ACM, 2014.

[137] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*, pages 263–272. Ieee, 2008.

[138] Furong Huang and Anima Anandkumar. Distributed latent dirichlet allocation on spark via tensor decomposition.

[139] Furong Huang, Sergiy Matusevych, Anima Anandkumar, Nikos Karampatziakis, and Paul Mineiro. Distributed latent dirichlet allocation via tensor factorization. In *NIPS Optimization Workshop*, 2014.

[140] Furong Huang, UN Niranjan, Mohammad Umar Hakeem, and Animashree Anandkumar. Fast detection of overlapping communities via online tensor methods. *arXiv preprint arXiv:1309.0787*, 2013.

[141] Heng Huang, Chris Ding, Dijun Luo, and Tao Li. Simultaneous tensor subspace selection and clustering: the equivalence of high order svd and k-means clustering. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge Discovery and Data mining*, pages 327–335. ACM, 2008.

[142] Ruitong Huang, Bing Xu, Dale Schuurmans, and Csaba Szepesvári. Learning with a strong adversary. *arXiv preprint arXiv:1511.03034*, 2015.

[143] Borbála Hunyadi, Daan Camps, Laurent Sorber, Wim Van Paesschen, Maarten De Vos, Sabine Van Huffel, and Lieven De Lathauwer. Block term decomposition for modelling epileptic seizures. *EURASIP Journal on Advances in Signal Processing*, 2014(1):1–19, 2014.

[144] Anh Huy Phan and Andrzej Cichocki. Parafac algorithms for large-scale problems. *Neurocomputing*, 74(11):1970–1984, 2011.

[145] Inah Jeon, Evangelos E Papalexakis, U Kang, and Christos Faloutsos. Haten2: Billion-scale tensor decompositions. In *2015 IEEE 31st International Conference on Data Engineering*, pages 1047–1058. IEEE, 2015.

[146] Meng Jiang, Alexander Beutel, Peng Cui, Bryan Hooi, Shiqiang Yang, and Christos Faloutsos. Spotting suspicious behaviors in multimodal data: A general metric and algortihms.

[147] Meng Jiang, Peng Cui, Fei Wang, Xinran Xu, Wenwu Zhu, and Shiqiang Yang. Fema: flexible evolutionary multi-faceted analysis for dynamic behavioral pattern discovery. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1186–1195. ACM, 2014.

[148] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 2019.

[149] U Kang, Evangelos E. Papalexakis, Abhay Harpale, and Christos Faloutsos. Gigatensor: scaling tensor analysis up by 100 times-algorithms and discoveries. In *ACM KDD'12*.

[150] Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 79–86, 2010.

[151] Henk AL Kiers and Albert Kinderen. A fast method for choosing the numbers of components in tucker3 analysis. *British Journal of Mathematical and Statistical Psychology*, 56(1):119–125, 2003.

[152] Mijung Kim and K Selçuk Candan. Decomposition-by-normalization (dbn): leveraging approximate functional dependencies for efficient tensor decomposition. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 355–364. ACM, 2012.

[153] Mijung Kim and Kasim Selçuk Candan. Approximate tensor decomposition within a tensor-relational algebraic framework. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 1737–1742. ACM, 2011.

[154] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[155] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *5th International Conference on Learning Representations (ICLR-17)*, 2017.

[156] J.M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632, 1999.

[157] Tamara G Kolda, Brett W Bader, and Joseph P Kenny. Higher-order web link analysis using multilinear algebra. In *Data Mining, Fifth IEEE International Conference on*, pages 8–pp. IEEE, 2005.

[158] Tamara G Kolda and Jimeng Sun. Scalable tensor decompositions for multi-aspect data mining. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 363–372. IEEE, 2008.

[159] T.G. Kolda and B.W. Bader. The tophits model for higher-order web link analysis. In *Workshop on Link Analysis, Counterterrorism and Security*, volume 7, pages 26–29, 2006.

[160] T.G. Kolda and B.W. Bader. Tensor decompositions and applications. *SIAM review*, 51(3), 2009.

[161] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434, 2008.

[162] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[163] Jean Kossaifi, Yannis Panagakis, Anima Anandkumar, and Maja Pantic. Tensorly: Tensor learning in python. *The Journal of Machine Learning Research*, 20(1):925–930, 2019.

[164] Alexander Kott, Ananthram Swami, and Patrick McDaniel. Six potential game-changers in cyber security: Towards priorities in cyber science and engineering. *arXiv preprint arXiv:1511.00509*, 2015.

[165] Danai Koutra, Evangelos E. Papalexakis, and Christos Faloutsos. Tensorsplat: Spotting latent anomalies in time. In *Informatics (PCI), 2012 16th Panhellenic Conference on*, pages 144–149. IEEE, 2012.

[166] Danai Koutra, Tai-You Ke, U Kang, Duen Horng Polo Chau, Hsing-Kuo Kenneth Pao, and Christos Faloutsos. Unifying guilt-by-association approaches: Theorems and fast algorithms. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 245–260. Springer, 2011.

[167] Pieter M Kroonenberg and Jan De Leeuw. Principal component analysis of three-mode data by means of alternating least squares algorithms. *Psychometrika*, 45(1):69–97, 1980.

[168] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

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

[170] Duc Trong Le, Hady W Lauw, and Yuan Fang. Basket-sensitive personalized item recommendation. IJCAI, 2017.

[171] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196, 2014.

[172] D.D. Lee and H.S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[173] Joonseok Lee, Seungyeon Kim, Guy Lebanon, Yoram Singer, and Samy Bengio. Llorma: Local low-rank matrix approximation. 2016.

[174] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the eleventh ACM*

*SIGKDD international conference on Knowledge discovery in data mining*, pages 177–187. ACM, 2005.

[175] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[176] Todd A. Letsche and Michael W. Berry. Large-scale information retrieval with latent semantic indexing. *Inf. Sci.*, 100(1-4):105–137, 1997.

[177] Bo Li, Yining Wang, Aarti Singh, and Yevgeniy Vorobeychik. Data poisoning attacks on factorization-based collaborative filtering. *Advances in neural information processing systems*, 29:1885–1893, 2016.

[178] Yifeng Li and Alioune Ngom. Classification of clinical gene-sample-time microarray expression data via tensor decomposition methods. In *Computational Intelligence Methods for Bioinformatics and Biostatistics*, pages 275–286. Springer, 2011.

[179] Dawen Liang, Jaan Altosaar, Laurent Charlin, and David M Blei. Factorization meets the item embedding: Regularizing matrix factorization with item co-occurrence. In *Proceedings of the 10th ACM conference on recommender systems*, pages 59–66, 2016.

[180] Chen Lin, Si Chen, Hui Li, Yanghua Xiao, Lianyun Li, and Qian Yang. Attacking recommender systems with augmented user profiles. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 855–864, 2020.

[181] Yu-Ru Lin, Jimeng Sun, Paul Castro, Ravi Konuru, Hari Sundaram, and Aisling Kelliher. Metafac: community discovery via relational hypergraph factorization. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 527–536. ACM, 2009.

[182] Ji Liu, Przemyslaw Musialski, Peter Wonka, and Jieping Ye. Tensor completion for estimating missing values in visual data. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(1):208–220, 2013.

[183] Charles F Van Loan. The ubiquitous kronecker product. *Journal of computational and applied mathematics*, 123(1):85–100, 2000.

[184] Bo Luo, Yannan Liu, Lingxiao Wei, and Qiang Xu. Towards imperceptible and robust adversarial example attacks against neural networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[185] Yi Luo and Henry Pfister. Adversarial defense of image classification using a variational auto-encoder. *arXiv preprint arXiv:1812.02891*, 2018.

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

[187] M.W. Mahoney, M. Maggioni, and P. Drineas. Tensor-cur decompositions for tensor-based data. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 327–336. ACM, 2006.

[188] Aditya Mantha, Yokila Arora, Shubham Gupta, Praveenkumar Kanumala, Zhiwei Liu, Stephen Guo, and Kannan Achan. A large-scale deep architecture for personalized grocery basket recommendations. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3807–3811. IEEE, 2020.

[189] Ching-Hao Mao, Chung-Jung Wu, Evangelos E. Papalexakis, Christos Faloutsos, and Tien-Cheu Kao. Malspot: Multi2 malicious network behavior patterns analysis. In *PAKDD'14*.

[190] K. Maruhashi, F. Guo, and C. Faloutsos. Multiaspectforensics: Pattern mining on large-scale heterogeneous networks with tensor analysis. In *Proceedings of the Third International Conference on Advances in Social Network Analysis and Mining*, 2011.

[191] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163, 2000.

[192] Shike Mei and Xiaojin Zhu. Using machine teaching to identify optimal training-set attacks on machine learners. In *AAAI*, pages 2871–2877, 2015.

[193] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On detecting adversarial perturbations. *arXiv preprint arXiv:1702.04267*, 2017.

[194] Saskia Metzler and Pauli Miettinen. Clustering boolean tensors. *arXiv preprint arXiv:1501.00696*, 2015.

[195] Pauli Miettinen. Boolean tensor factorizations. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 447–456. IEEE, 2011.

[196] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[197] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[198] Tom M Mitchell, Svetlana V Shinkareva, Andrew Carlson, Kai-Min Chang, Vicente L Malave, Robert A Mason, and Marcel Adam Just. Predicting human brain activity associated with the meanings of nouns. *science*, 320(5880):1191–1195, 2008.

[199] Andriy Mnih and Russ R Salakhutdinov. Probabilistic matrix factorization. *Advances in neural information processing systems*, 20:1257–1264, 2007.

[200] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 86–94. Ieee, 2017.

[201] Morten Mørup and Lars Kai Hansen. Automatic relevance determination for multi-way models. *Journal of Chemometrics*, 23(7-8):352–363, 2009.

[202] Morten Mørup, Lars Kai Hansen, and Kristoffer Hougaard Madsen. Modeling latency and shape changes in trial based neuroimaging data. In *Signals, Systems and Computers (ASILOMAR), 2011 Conference Record of the Forty Fifth Asilomar Conference on*, pages 439–443. IEEE, 2011.

[203] Yang Mu, Wei Ding, Melissa Morabito, and Dacheng Tao. Empirical discriminative tensor analysis for crime forecasting. In *Knowledge Science, Engineering and Management*, pages 293–304. Springer, 2011.

[204] H Neudecker. A note on kronecker matrix products and matrix equation systems. *SIAM Journal on Applied Mathematics*, 17(3):603–606, 1969.

[205] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *Icml*, 2011.

[206] D. Nion and N.D. Sidiropoulos. Adaptive algorithms to track the parafac decomposition of a third-order tensor. *Signal Processing, IEEE Transactions on*, 57(6):2299–2310, 2009.

[207] Dimitri Nion, Kleanthis N Mokios, Nicholas D Sidiropoulos, and Alexandros Potamianos. Batch and adaptive parafac-based blind separation of convolutive speech mixtures. *Audio, Speech, and Language Processing, IEEE Transactions on*, 18(6):1193–1207, 2010.

[208] Ivan V Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.

[209] D O'Halloran and E Kvochko. Industrial internet of things: Unleashing the potential of connected products and services. In *World Economic Forum*, page 40, 2015.

[210] R. Pang, M. Allman, M. Bennett, J. Lee, V. Paxson, and B. Tierney. A first look at modern enterprise traffic. In *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, pages 2–2. USENIX Association, 2005.

[211] S. Papadimitriou, J. Sun, and C. Faloutsos. Streaming pattern discovery in multiple time-series. In *Proceedings of the 31st international conference on Very large data bases*, pages 697–708. VLDB Endowment, 2005.

[212] E.E. Papalexakis, C. Faloutsos, and N.D. Sidiropoulos. Tensors for data mining and data fusion: Models, applications, and scalable algorithms. *ACM Trans. on Intelligent Systems and Technology.*

[213] E.E. Papalexakis and N.D. Sidiropoulos. Co-clustering as multilinear decomposition with sparse latent factors. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 2064–2067. IEEE, 2011.

[214] Evangelos E. Papalexakis. Automatic unsupervised tensor mining with quality assessment. *Working Paper*, 2015.

[215] Evangelos E. Papalexakis, Tudor Dumitras, Duen Horng Chau, B Aditya Prakash, and Christos Faloutsos. Sharkfin: Spatio-temporal mining of software adoption and penetration. *Social Network Analysis and Mining*, 4(1):1–15, 2014.

[216] Evangelos E. Papalexakis, Christos Faloutsos, and Nicholas D. Sidiropoulos. Parcube: Sparse parallelizable candecomp-parafac tensor decomposition. *ACM Trans. Knowl. Discov. Data*, 10(1):3:1–3:25, July 2015.

[217] Evangelos E Papalexakis, Christos Faloutsos, and Nicholas D Sidiropoulos. Tensors for data mining and data fusion: Models, applications, and scalable algorithms. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(2):16, 2017.

[218] Evangelos E Papalexakis, Tom M Mitchell, Nicholas D Sidiropoulos, Christos Faloutsos, Partha Pratim Talukdar, and Brian Murphy. Turbo-smt: Parallel coupled sparse matrix-tensor factorizations and applications. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 2016.

[219] Papalexakis, Evangelos E. Automatic unsupervised tensor mining with quality assessment. In *SIAM SDM*, 2016.

[220] Nicolas Papernot, Nicholas Carlini, Ian Goodfellow, Reuben Feinman, Fartash Faghri, Alexander Matyasko, Karen Hambardzumyan, Yi-Lin Juang, Alexey Kurakin, and Ryan Sheatsley. cleverhans v2. 0.0: an adversarial machine learning library. *arXiv preprint arXiv:1610.00768*, 2016.

[221] Nicolas Papernot, Fartash Faghri, Nicholas Carlini, Ian Goodfellow, Reuben Feinman, Alexey Kurakin, Cihang Xie, Yash Sharma, Tom Brown, Aurko Roy, et al. Technical report on the cleverhans v2. 1.0 adversarial examples library. *arXiv preprint arXiv:1610.00768*, 2016.

[222] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, pages 372–387. IEEE, 2016.

[223] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 582–597. IEEE, 2016.

[224] R. Penrose. A generalized inverse for matrices. In *Proc. Cambridge Philos. Soc*, volume 51, pages 406–413. Cambridge Univ Press, 1955.

[225] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM, 2014.

[226] Bryan Perozzi, Vivek Kulkarni, and Steven Skiena. Walklets: Multiscale graph embeddings for interpretable network classification. *arXiv preprint arXiv:1605.02115*, 2016.

[227] Ioakeim Perros, Robert Chen, Richard Vuduc, and Jimeng Sun. Sparse hierarchical tucker factorization and its application to healthcare. In *Data Mining (ICDM), 2015 IEEE 15th International Conference on*. IEEE, 2015.

[228] Trang Pham, Truyen Tran, Dinh Q Phung, and Svetha Venkatesh. Column networks for collective classification. In *AAAI*, pages 2485–2491, 2017.

[229] A.H. Phan and A. Cichocki. Block decomposition for very large-scale nonnegative tensor factorization. In *Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 2009 3rd IEEE International Workshop on*, pages 316–319. IEEE, 2009.

[230] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[231] Steffen Rendle and Lars Schmidt-Thieme. Pairwise interaction tensor factorization for personalized tag recommendation. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 81–90. ACM, 2010.

[232] Yan Rong, Sun Jimeng, and Spiros Papadimitriou. Large-scale data mining: Mapreduce and beyond. `http://yanrong.info/`.

[233] Bita Rouhani, Mohammad Samragh, Tara Javidi, Farinaz Koushanfar, et al. Safe machine learning and defeating adversarial attacks. *IEEE Security and Privacy (S&P) Magazine*, 2018.

[234] Sebnem Rusitschka and Edward Curry. Big data in the energy and transport sectors. In *New Horizons for a Data-Driven Economy*, pages 225–244. Springer, 2016.

[235] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295, 2001.

[236] Aaron Schein, John Paisley, David M Blei, and Hanna Wallach. Bayesian poisson tensor factorization for inferring multilateral relations from sparse dyadic event counts. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1045–1054. ACM, 2015.

[237] Neil Shah, Alex Beutel, Brian Gallagher, and Christos Faloutsos. Spotting suspicious link behavior with fbox: An adversarial perspective. In *Data Mining (ICDM), 2014 IEEE International Conference on*, pages 959–964. IEEE, 2014.

[238] Neil Shah, Alex Beutel, Brian Gallagher, and Christos Faloutsos. Spotting suspicious link behavior with fbox: An adversarial perspective. *arXiv preprint arXiv:1410.3915*, 2014.

[239] N Sidiropoulos, Evangelos E. Papalexakis, and C Faloutsos. Parallel randomly compressed cubes: A scalable distributed architecture for big tensor decomposition. *IEEE Signal Processing Magazine*, 2014.

[240] N.D. Sidiropoulos, G.B. Giannakis, and R. Bro. Blind parafac receivers for ds-cdma systems. *Signal Processing, IEEE Transactions on*, 48(3):810–823, 2000.

[241] ND Sidiropoulos, Papalexakis, Evangelos E, and C Faloutsos. A parallel algorithm for big tensor decomposition using randomly compressed cubes (paracomp). In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, 2014.

[242] Nicholas D Sidiropoulos, Lieven De Lathauwer, Xiao Fu, Kejun Huang, Evangelos E Papalexakis, and Christos Faloutsos. Tensor decomposition for signal processing and machine learning. *IEEE Signal Processing Magazine*.

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

[244] Sucheta Soundarajan, Acar Tamersoy, Elias B Khalil, Tina Eliassi-Rad, Duen Horng Chau, Brian Gallagher, and Kevin Roundy. Generating graph snapshots from streaming edge data. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 109–110. International World Wide Web Conferences Steering Committee, 2016.

[245] Alwin Stegeman and Pierre Comon. Subtracting a best rank-1 approximation may increase tensor rank. *Linear Algebra and its Applications*, 433(7):1276 – 1300, 2010.

[246] Rajmonda Sulo, Tanya Berger-Wolf, and Robert Grossman. Meaningful selection of temporal resolution for dynamic networks. In *Proceedings of the Eighth Workshop on Mining and Learning with Graphs*, pages 127–136. ACM, 2010.

[247] J. Sun, S. Papadimitriou, C.Y. Lin, N. Cao, S. Liu, and W. Qian. Multivis: Content-based social network exploration through multi-way visual analysis. In *Proc. SDM*, volume 9, pages 1063–1074, 2009.

[248] J. Sun, D. Tao, and C. Faloutsos. Beyond streams and graphs: dynamic tensor analysis. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 374–383. ACM, 2006.

[249] Jimeng Sun, Dacheng Tao, Spiros Papadimitriou, Philip S Yu, and Christos Faloutsos. Incremental tensor analysis: Theory and applications. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 2(3):11, 2008.

[250] Jimeng Sun, Charalampos E. Tsourakakis, Evan Hoke, Christos Faloutsos, and Tina Eliassi-Rad. Two heads better than one: pattern discovery in time-evolving multi-aspect data. *Data Mining and Knowledge Discovery*, 17(1):111–128, 2008.

[251] J.T. Sun, H.J. Zeng, H. Liu, Y. Lu, and Z. Chen. Cubesvd: a novel approach to personalized web search. In *Proceedings of the 14th international conference on World Wide Web*, pages 382–390. ACM, 2005.

[252] Mingjie Sun, Jian Tang, Huichen Li, Bo Li, Chaowei Xiao, Yao Chen, and Dawn Song. Data poisoning attack against unsupervised node embedding methods. *arXiv preprint arXiv:1810.12881*, 2018.

[253] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S Yu, and Tianyi Wu. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *VLDB*, 2011.

[254] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

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

[256] Jiaxi Tang, Hongyi Wen, and Ke Wang. Revisiting adversarially learned injection attacks against recommender systems. In *Fourteenth ACM Conference on Recommender Systems*, pages 318–327, 2020.

[257] Jinhui Tang, Xiaoyu Du, Xiangnan He, Fajie Yuan, Qi Tian, and Tat-Seng Chua. Adversarial training towards robust multimedia recommender system. *IEEE Transactions on Knowledge and Data Engineering*, 32(5):855–867, 2019.

[258] Dacheng Tao, Xuelong Li, Weiming Hu, Stephen Maybank, and Xindong Wu. Supervised tensor learning. In *Data Mining, Fifth IEEE International Conference on*, pages 8–pp. IEEE, 2005.

[259] Dacheng Tao, Mingli Song, Xuelong Li, Jialie Shen, Jimeng Sun, Xindong Wu, Christos Faloutsos, and Stephen J Maybank. Bayesian tensor approach for 3-d face modeling. *Circuits and Systems for Video Technology, IEEE Transactions on*, 18(10):1397–1410, 2008.

[260] Dacheng Tao, Jimeng Sun, Xindong Wu, Xuelong Li, Jialie Shen, Stephen J Maybank, and Christos Faloutsos. Probabilistic tensor analysis with akaike and bayesian information criteria. In *International Conference on Neural Information Processing*, pages 791–801. Springer, 2007.

[261] Giorgio Tomasi and Rasmus Bro. Parafac and missing values. *Chemometrics and Intelligent Laboratory Systems*, 75(2):163–180, 2005.

[262] Giorgio Tomasi and Rasmus Bro. A comparison of algorithms for fitting the parafac model. *Computational Statistics & Data Analysis*, 50(7):1700–1734, 2006.

[263] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

[264] C.E. Tsourakakis. Mach: Fast randomized tensor decompositions. *Arxiv preprint arXiv:0909.4969*, 2009.

[265] L.R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.

[266] M. Vasilescu and D. Terzopoulos. Multilinear analysis of image ensembles: Tensorfaces. *Computer Vision ECCV 2002*, pages 447–460, 2002.

[267] Bimal Viswanath, Alan Mislove, Meeyoung Cha, and Krishna P. Gummadi. On the evolution of user interaction in facebook. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Social Networks (WOSN'09)*, August 2009.

[268] Mengting Wan, Di Wang, Jie Liu, Paul Bennett, and Julian McAuley. Representing and recommending shopping baskets with complementarity, compatibility and loyalty. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 1133–1142, 2018.

[269] Qinyong Wang, Hongzhi Yin, Zhiting Hu, Defu Lian, Hao Wang, and Zi Huang. Neural memory streaming recommender networks with adversarial training. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2467–2475, 2018.

[270] Yilun Wang, Yu Zheng, and Yexiang Xue. Travel time estimation of a path using sparse trajectories. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 25–34, New York, NY, USA, 2014. ACM.

[271] Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan L Yuille, and Kaiming He. Feature denoising for improving adversarial robustness. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 501–509, 2019.

[272] Liang Xiong, Xi Chen, Tzu-Kuo Huang, Jeff G Schneider, and Jaime G Carbonell. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *SDM*, volume 10, pages 211–222. SIAM, 2010.

[273] Rong Yan. Large-scale data mining challenge in facebook". `http://yanrong.info/`.

[274] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y Chang. Network representation learning with rich text information. In *IJCAI*, pages 2111–2117, 2015.

[275] Jing Yuan, Yu Zheng, Xing Xie, and Guangzhong Sun. Driving with knowledge from the physical world. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 316–324. ACM, 2011.

[276] Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. Adversarial examples: Attacks and defenses for deep learning. *IEEE transactions on neural networks and learning systems*, 30(9):2805–2824, 2019.

[277] Fuzheng Zhang, Nicholas Jing Yuan, David Wilkie, Yu Zheng, and Xing Xie. Sensing the pulse of urban refueling behavior: A perspective from taxi mobility. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 6(3):37, 2015.

[278] Q. Zhang, M. Berry, B. Lamb, and T. Samuel. A parallel nonnegative tensor factorization algorithm for mining global climate data. *Computational Science–ICCS 2009*, pages 405–415, 2009.

[279] Yongfeng Zhang, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. Understanding the sparsity: Augmented matrix factorization with sampled constraints on unobservables. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1189–1198. ACM, 2014.

[280] Q Zhao, L Zhang, and A Cichocki. Bayesian cp factorization of incomplete tensors with automatic rank determination.

[281] Zhe Zhao, Zhiyuan Cheng, Lichan Hong, and Ed H Chi. Improving user topic interest profiles by behavior factorization. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1406–1416. International World Wide Web Conferences Steering Committee, 2015.

[282] Shandian Zhe, Yuan Qi, Youngja Park, Ian Molloy, and Suresh Chari. Dintucker: Scaling up gaussian process models on multidimensional arrays with billions of elements. *arXiv preprint arXiv:1311.2663*, 2013.

[283] Vincent W Zheng, Yu Zheng, Xing Xie, and Qiang Yang. Towards mobile intelligence: Learning from gps history data for collaborative recommendation. *Artificial Intelligence*, 184:17–37, 2012.

[284] Vincent Wenchen Zheng, Bin Cao, Yu Zheng, Xing Xie, and Qiang Yang. Collaborative filtering meets mobile recommendation: A user-centered approach. In *AAAI*, volume 10, pages 236–241, 2010.

[285] Yu Zheng, Tong Liu, Yilun Wang, Yanmin Zhu, Yanchi Liu, and Eric Chang. Diagnosing new york city's noises with ubiquitous data. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 715–725. ACM, 2014.

[286] Ziwei Zhu, Xia Hu, and James Caverlee. Fairness-aware tensor-based recommendation. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 1153–1162, 2018.

[287] Sonja Zillner, Tilman Becker, Ricard Munné, Kazim Hussain, Sebnem Rusitschka, Helen Lippell, Edward Curry, and Adegboyega Ojo. Big data-driven innovation in industrial sectors. In *New Horizons for a Data-Driven Economy*, pages 169–178. Springer, 2016.

[288] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2847–2856. ACM, 2018.