Explaining the Vulnerabilities of Machine Learning through Visual Analytics

by

Tiankai Xie

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved July 2023 by the
Graduate Supervisory Committee:

Ross Maciejewski, Chair
Huan Liu
Chris Bryan
Hanghang Tong

ARIZONA STATE UNIVERSITY

August 2023

ABSTRACT

Machine learning models are increasingly being deployed in real-world applications where their predictions are used to make critical decisions in a variety of domains. The proliferation of such models has led to a burgeoning need to ensure the reliability and safety of these models, given the potential negative consequences of model vulnerabilities. The complexity of machine learning models, along with the extensive data sets they analyze, can result in unpredictable and unintended outcomes. Model vulnerabilities may manifest due to errors in data input, algorithm design, or model deployment, which can have significant implications for both individuals and society. To prevent such negative outcomes, it is imperative to identify model vulnerabilities at an early stage in the development process. This will aid in guaranteeing the integrity, dependability, and safety of the models, thus mitigating potential risks and enabling the full potential of these technologies to be realized. However, enumerating vulnerabilities can be challenging due to the complexity of the real-world environment. Visual analytics, situated at the intersection of human-computer interaction, computer graphics, and artificial intelligence, offers a promising approach for achieving high interpretability of complex black-box models, thus reducing the cost of obtaining insights into potential vulnerabilities of models. This research is devoted to designing novel visual analytics methods to support the identification and analysis of model vulnerabilities. Specifically, generalizable visual analytics frameworks are instantiated to explore vulnerabilities in machine learning models concerning security (adversarial attacks and data perturbation) and fairness (algorithmic bias). In the end, a visual analytics approach is proposed to enable domain experts to explain and diagnose the model improvement of addressing identified vulnerabilities of machine learning models in a human-in-the-loop fashion. The proposed methods hold the potential to enhance the security and fairness of machine learning models deployed in critical real-world applications.

DEDICATION

To my wife, Jienan Wang, and our family, for their selfless love, unwavering support, and constant encouragement. Their influence has made me a better person, and I am eternally grateful for their presence in my life.

ACKNOWLEDGMENTS

I would like to extend my sincere appreciation to my advisor, Dr. Ross Maciejewski, for his invaluable support and guidance throughout my Ph.D. journey. Dr. Maciejewski's unwavering expertise and encouragement have been instrumental in shaping my research, and I am grateful for his mentorship. His passion for research, attention to detail, and rigorous standards have been a source of inspiration for me, and I am honored to have had the opportunity to work under his guidance.

I would also like to express my gratitude to Dr. Yuxin Ma, my postdoctoral mentor, for his invaluable guidance and support at the outset of my research. Dr. Ma's deep understanding of the field and his thoughtful insights have been instrumental in setting me on the right track. His encouragement and support throughout my research journey have been essential to my success.

I would like to acknowledge my wife, Jienan Wang, my parents Min Tian and Yongqing Xie, as well as all of my family members for their unfailing support and encouragement throughout my doctoral journey. Their patience, selflessness, and love have sustained me during the challenging moments, and I am deeply grateful for their presence in my life. Their belief in me and my research has been a source of inspiration and motivation, and I feel incredibly fortunate to have their support. I recognize the sacrifices they have made to support me, and I am grateful for their endless encouragement, care, and love.

Lastly, I express my heartfelt appreciation to all of my colleagues, lab mates, and friends who have accompanied me throughout this journey. Their insights, support, and contributions have been priceless in achieving success in this endeavor, and I am profoundly thankful for their consistent dedication. Their expertise and commitment have made many research projects possible, and I am privileged to have collaborated and worked with them. I am excited to continue this partnership with them in the foreseeable future.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

Figure Page

Chapter 1

INTRODUCTION

In the era of Big Data, Artificial Intelligence and Machine Learning have made immense strides in developing models and classifiers for real-world phenomena. To date, applications of these models are found in cancer diagnosis tools (Esteva et al. 2017), self-driving cars (Martinez et al. 2018), biometrics (Sundararajan and Woodard 2018), and numerous other areas. Unfortunately, the real-world application of these models introduces a dynamic environment where vulnerabilities introduced during model development can have unintended consequences.

Consider e-mail spam filtering as an example. To date, a variety of machine learning methods (Blanzieri and Bryl 2008; Caruana and Li 2012) have been developed to protect e-mail inboxes from unwanted messages. These methods build models to classify e-mail as spam or not-spam. However, adversaries still want their spam messages to reach your inbox, and these adversaries try to build input data (i.e., spam e-mails) that will fool the model into classifying their spam as safe. This can be done by misspelling words that might cause the machine learning classifier to flag a mail as spam or by inserting words and phrases that might cause the classifier to believe the message is safe. Other adversarial attacks have explored methods to fake bio-metric data to gain access to personal accounts (Biggio et al. 2015) and to cause computer vision algorithms to misclassify stop signs (S.-T. Chen et al. 2018). Such exploits can have devastating effects, and researchers are finding that applications of machine learning in real-world environments are increasingly vulnerable to adversarial attacks. As such, it is imperative that model designers be able to diagnose security risks in their machine learning models.

Such risks are not only posed from attackers, but can also come in the form of legal challenges to machine learning algorithms that exhibit bias. Algorithmic fairness has become increasingly important in data mining and machine learning. This has led to a proliferation of algorithmic enhancements to address potential fairness issues that can occur in black-box models. Although researchers have been developing methods to guarantee the fairness of data-driven models (Q. Wang et al. 2021; Cabrera et al. 2019; Ahn and Lin 2020; Rahman et al. 2019; Yao and Huang 2017; Tsioutsiouliklis et al. 2020; J. Kang et al. 2020; Bose and Hamilton 2019; Zehlike et al. 2017), it has been reported that biases can still be observed even after the fairness algorithms are applied (Sühr, Hilgard, and Lakkaraju 2020).

Take for example legal definitions of fairness that focus on gender and ethnicity attributes of the data. Here, numerous algorithms have been proposed to correct for single attribute biases. However, as noted by Wang et al. (Q. Wang et al. 2021), algorithms might be subject to *indirect discrimination*, where a protected class attribute might be correlated to another feature in the dataset, for example, location attributes such as ZIP Code might have implicit racial information as the distribution of ethnicity is geographically unbalanced. Thus, fairness solutions that only adjust for a single data attribute can still suffer from algorithmic biases. Given such issues, it is difficult to balance algorithmic results under potentially conflicting definitions of fairness, and recent work (Friedler, Scheidegger, and Venkatasubramanian 2021; Kleinberg, Mullainathan, and Raghavan 2016; Cabrera et al. 2019) has even discussed an *impossibility theorem* for fairness noting that it may be impossible to guarantee fairness that satisfies all constraints.

Such challenges reflect the vulnerabilities of machine learning models in different aspects. In this proposal, those challenges are identified as two main categories of machine learning vulnerabilities: *machine learning security* and *machine learning fairness*, where the former depicts the vulnerability that models can potentially be sabotaged by either the nature of models themselves or ingenue data, and the latter describes the vulnerability that the models are misled to perform societal tasks with different sorts of biases.

1.1 Problem Statement

The underlying reason for machine learning models to have such vulnerabilities is often related to their low interpretability. Since many models are treated as black boxes, it is difficult to explain how and why predictions are made, and many predictions can be caused by different factors. As such, if predictions go wrong, researchers and developers may be unable to diagnose the root causes, which raises the difficulty of improving the models. In this proposal, detailed aspects of vulnerabilities are described with respect to security and fairness.

1.1.1 Security in Machine Learning

From a security perspective, models are considered vulnerable when they underperform due to perturbations of either training data or testing data. According to Huang et al. (Huang et al. 2011) and Vorobeychik et al. (Vorobeychik and Kantarcioglu 2018b), such methods are defined as *adversarial attacks*. Attack methods based on the perturbing the training data are referred to as poisoning attacks, and those based on modifying the test data are evasion attacks. Poisoning attacks usually manipulate the training data by flipping the labels or adding obscure instances with wrong labels to confuse the model, such that the random/targeted instances are mispredicted. Evasion attacks try to craft test instances that look normal to humans while models can hardly recognize. Both attacks leverage the vulnerabilities of models to output unexpected results. Thus, it is necessary to proactively defend against the attack strategies and support model developers in understanding how attacks can compromise machine learning models.

1.1.2 Fairness in Machine Learning

In addition to security, machine learning models are vulnerable to biases that render their decisions “unfair”. In the context of decision-making, fairness is the absence of any prejudice or favoritism toward an individual or a group based on their inherent or acquired characteristics. Thus, an unfair model is one whose decisions are skewed toward a particular group of people (Mehrabi et al. 2019). In addition to explicit societal bias derived from the real-world data to the models, implicit or indirect biases may also be generated from model debiasing. Researchers have recently proposed many definitions of fairness. *Group fairness* and *individual fairness* are the most popular definitions, where group fairness focuses on whether or not members of a protected class have the same probability of being assigned a positive outcome and individual fairness focuses on whether similar individuals are treated consistently. However, these definitions are sometimes conflicting, meaning that if group fairness is guaranteed, individual fairness is sometimes violated and vice versa. Therefore, tools for diagnosing and reasoning with potential unfairness or bias in the development of machine learning models are critical.

1.1.3 Robustness in Machine Learning

Finally, the goal of revealing varieties of vulnerabilities of machine learning models is to understand their deficiencies and improve the robustness of the model. Thus, it is necessary to build the tool to leverage the revealed vulnerabilities and improve them. With the advantages of visual analytics techniques, it seems feasible to develop interpretable, interactive, transparent system to help users with different knowledge background to understand the black-box models and build more robust model.

1.2 Aim of the Work

Such challenges seem to necessitate a human-in-the-loop approach, where analysts can audit their machine learning algorithms for potential vulnerabilities and reveal the potential deficiencies of the model. This work aims to address and explore machine learning vulnerabilities with respect to machine learning security and machine learning fairness through visual analytics, and then enhance the robustness of machine learning models based on them. For machine learning security, novel visual analytics frameworks are developed to explore vulnerabilities of models under adversarial attack algorithms and data perturbations. For machine learning fairness, another visual analytics framework is proposed to address the group and individual fairness issues in graph mining models. The final stage of the dissertation is to explore how visual analytics can move from explanations of why a vulnerability occurred to also suggesting potential mechanisms to overcome identified vulnerabilities.

1.3 Outline and Individual Contributions

The rest of the content is organized as follows. Chapter 2 describes the state-of-art progress in terms of vulnerability-related work. Chapters 3 and 4 introduce the main framework workflow and visualization designs of the proposed work. Finally, chapter 5 discusses the robust analysis of machine learning models. The following list contains the published works and contributions as a means of documenting current progress toward the dissertation.

1. A visual analytics framework for explaining vulnerabilities to adversarial machine learning (Ma et al. 2020), contributed as the second author and was responsible for system implementation and data processing (note that for this paper, the first and second authors were considered to be equivalent);
2. Visual analytics methods for auditing the sensitivity of graph-based ranking (Tiankai Xie, Yuxin Ma, Hanghang Tong, et al. 2021), contributed as the first author and was responsible for data processing, system implementation and paper writing;
3. A visual analytics framework for exploring algorithmic fairness in graph mining models (Xie, Ma, Kang, et al. 2021), contributed as the first author and was responsible for data processing, system implementation and paper writing.
4. A visual analytics framework for validating machine learning models, contributed as the first author and was responsible for data processing, system implementation and paper writing.
5. A visual analytics framework for validating data augmentation, contributed as the first author and was responsible for data processing, system implementation and paper writing.

Chapter 2

LITERATURE REVIEW

This chapter introduces related work in machine learning vulnerability and visual analytics. The first section introduces the state-of-the-art methods for machine learning security analysis, the second section introduces the machine learning fairness analysis, and the third section introduces the visual analytics methods for machine learning robustness. Each section also contains the related work of tasks mentioned above in visual analytics.

2.1 Secure Machine Learning

Adversarial machine learning (AML) is a rapidly growing field that focuses on studying the security and vulnerability of machine learning models. AML aims to identify and exploit the weaknesses of machine learning models through the deliberate introduction of malicious inputs, known as adversarial attacks. These attacks can cause a model to make incorrect predictions or even compromise the integrity of the model itself. The need for AML arises due to the growing prevalence of machine learning systems in critical applications such as finance, healthcare, and transportation, where a failure in the model can have significant consequences. Researchers in the AML field work to develop new techniques for detecting and defending against adversarial attacks, ultimately aiming to create more robust and secure machine learning models.

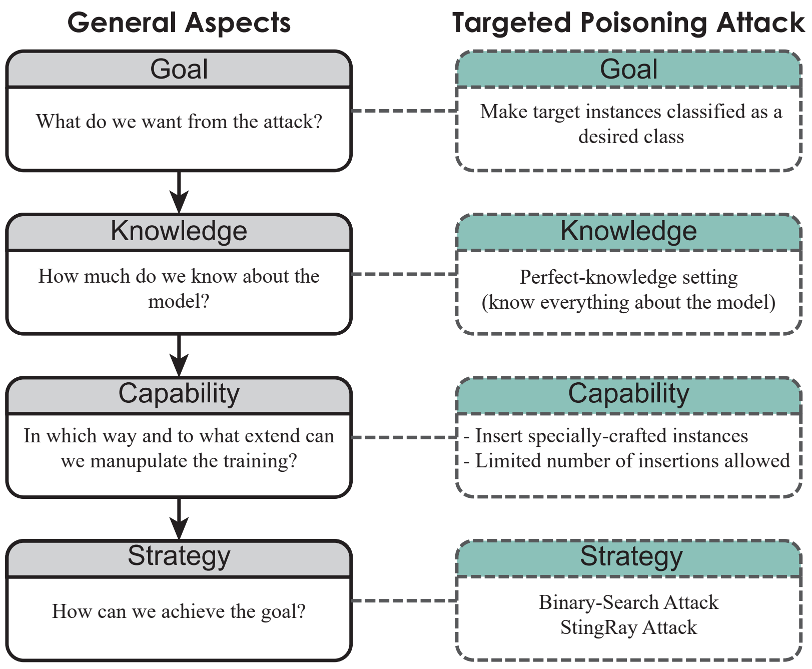

General Aspects
Targeted Poisoning Attack
Goal
What do we want from the attack?
Goal
Make target instances classified as a desired class
Knowledge
How much do we know about the model?
Knowledge
Perfect-knowledge setting (know everything about the model)
Capability
In which way and to what extend can we manupulate the training?
Capability
- Insert specially-crafted instances
- Limited number of insertions allowed
Strategy
How can we achieve the goal?
Strategy
Binary-Search Attack
StingRay Attack

Figure 1. Key features of an adversary. (Left) The general components an adversary must consider when planning an attack. (Right) Specific considerations in a data poisoning attack.

2.1.1 Adversarial Machine Learning

Since our goal is to support the exploration of model vulnerabilities, it is critical to identify common attack strategies and model weaknesses. The four main features of an adversary (or attacker) (Vorobeychik and Kantarcioglu 2018a; Biggio and Roli 2018) are the adversary's Goal, Knowledge, Capability, and Strategy, Figure 1 (Left).

Goal: In adversarial machine learning, an attacker's goal can be separated into two major categories: *targeted attacks* and *reliability attacks*. In a targeted attack, the attacker seeks to insert specific malicious instances or regions in the input feature space and prevent these insertions from being detected (Yingqi Liu et al. 2018; Shafahi et al. 2018). In a reliability attack, the goal of the attacker is to maximize the overall prediction error of the model and make the model unusable for making predictions (Steinhardt, Koh, and Liang 2017).

Knowledge: The information that can be accessed by an attacker plays a significant role in how an attacker will design and deploy attack operations. The more knowledge an attacker has about a model (victim), the more precise an attack can be. In a *black-box* model, the attacker will have imprecise (or even no) knowledge about the machine learning model, while in a *white-box* setting, the attacker will have most (if not all) of the information about the model, including the model type, hyper-parameters, input features, and training dataset (Biggio and Roli 2018).

Capability: The capability of the attacker refers to when and what the attacker can do to influence the training and testing process to achieve the attack's goal. Where the attack takes place (i.e., the stage of the modeling process - training, testing) limits the capability of the attacker. For example, *poisoning attacks* (Xiao, Xiao, and Eckert 2012; Biggio, Nelson, and Laskov 2012) take place during the training-stage, and the attacker attempts to manipulate the training dataset. Typical operations in data poisoning attacks include adding noise instances and flipping labels of existing instances. An *evasion attack* (Dalvi et al. 2004; Biggio et al. 2013; Goodfellow, Shlens, and Szegedy 2015) takes place during the testing stage. Such an attack is intended to manipulate unlabeled data in order to avoid detection in the testing stage without touching the training process. In all of these cases, the attacker is constrained by how much they can manipulate either the training or test data without being detected or whether the training and test data are even vulnerable to such attacks.

Strategy: Given the attacker's goal, knowledge, and capabilities, all that remains is for the attacker to design an attack strategy. An optimal attack strategy can be described as maximizing the attack effectiveness while minimizing the cost of data manipulation or other constraints (Mei and Zhu 2015).

Currently, numerous adversarial machine learning attacks are being developed, with evasion and poisoning strategies receiving the most attention (Thomas and Tabrizi 2018). In evasion attacks, a common strategy is to add noise to test data instances. Goodfellow et al. (Goodfellow, Shlens, and Szegedy 2015) proposed a method to add "imperceptible" noise to an image, which can drastically confuse a trained deep neural network resulting in unwanted predictions. For poisoning attacks, the strategies are usually formalized as bi-level optimization problems, such as gradient ascending (Biggio, Nelson, and Laskov 2012) and machine teaching (Mei and Zhu 2015). Common among these attacks is the goal of manipulating the trained model, and it is critical for users to understand where and how their models may be vulnerable.

2.1.2 Visual Analytics for Secure Machine Learning

The uptake of many machine learning systems has been hampered by their inherent black-box nature (Krause, Perer, and Ng 2016). Users want to know why models perform a certain way, why models make specific decisions, and why models succeed or fail in specific instances (Endert et al. 2017). The visual analytics community has tackled this problem by developing methods to open the black-box of machine learning models (Bertini and Lalanne 2009; S. Liu et al. 2017; Y. Lu et al. 2017; J. Lu et al. 2017). The goal is to improve the explainability of models, allow for more user feedback, and increase the user's trust in a model. To date, a variety of visual analytics methods have been developed to support model explainability and performance diagnosis.

While the visual analytics community has focused on explainability with respect to model input-outputs, hidden layers, underlying "black-box" mechanisms, and performance metrics less work has focused on explaining model vulnerabilities. Liu et al. (M. Liu et al. 2018) present AEVis, a visual analytics tool for deep learning models, which visualizes data-paths along with the hidden layers in order to interpret the prediction process of adversarial examples. However, the approach is tightly coupled with generating adversarial examples for deep neural networks, which is not extensible to other attack forms and model types. My work builds upon previous visual analytics explainability work, adopting coordinated multiple views that support various types of models and attack strategies. What is unique in our work is the integration of attack strategies into the visual analytics pipeline to highlight model vulnerabilities.

2.2 Fair Machine Learning

Machine learning fairness is a growing area of research that aims to ensure that machine learning algorithms do not produce biased or discriminatory outcomes. Machine learning algorithms can learn and amplify biases present in the data they are trained on, which can have real-world consequences on individuals or groups. For example, biased algorithms could result in unfair decisions such as rejecting a loan application based on an individual's race or gender. The study of machine learning fairness involves identifying and mitigating sources of bias in the data and the algorithm itself, and ensuring that the algorithm is not unfairly discriminating against certain groups. As machine learning is increasingly used in decision-making processes, it is crucial to ensure that these algorithms are fair and do not perpetuate harmful biases.

2.2.1 Fairness in Machine Learning

In order to account for potential algorithmic bias, numerous iterations of fairness-aware graph mining algorithms have been developed focusing on individual fairness and group fairness. Kamishima et al (Kamishima, Akaho, and Asoh 2012) employ regularization-based collaborative filtering which minimizes the average ratings among different groups to control for potential bias. In graph-based clustering, Kleindessner et al. (Kleindessner et al. 2019) propose a fairness notion to balance the number of elements in each cluster based on different demographic groups. Bose et al. (Bose and Hamilton 2019) employ an adversarial framework to achieve statistical parity for the learned embedding results across sensitive attributes. Kang et al. (J. Kang et al. 2020) study the individual fairness problem in graph mining models and propose an optimization-based framework for diagnosing and debiasing graph mining models by three individual approaches: debiasing data, debiasing model as well as debiasing result. However, these approaches only guarantee group fairness or individual fairness without considering whether applying constraints for group fairness affects individual fairness or vice versa. As such, tools that can support fairness auditing between variations of graph mining algorithms are critical for identifying algorithmic fairness.

2.2.2 Visual Analytics for Fair Machine Learning

Model explainability is also highly coupled to issues of algorithmic fairness. Given the fact that definitions of fairness can be highly task-dependent, recent work in the visual analytics community has begun exploring methods for human-in-the-loop fairness auditing and exploration. Cabrera et al. (Cabrera et al. 2019) propose a visual analytics framework (FairVis) for discovering intersectional bias by inspecting machine learning models' performance on different groups, where a group is defined with respect to a set of potential sensitive attributes. The analyst can select the performance metric, e.g., accuracy, F1 score, true positive rate, etc. Ahn et al. (Ahn and Lin 2020) propose a general visual analytics framework (FairSight) for diagnosing the fairness of top-k ranking results by considering both nodes and groups. The framework provides metrics for diagnosing both individual fairness and group fairness in terms of a single sensitive attribute. However, multi-attribute fairness diagnosis was unexplored. Wang et al. (DiscriLens) (Q. Wang et al. 2021) also investigated issues of fairness in classification tasks by visualizing the unbalanced proportion between user-defined groups with respect to a single sensitive attribute. These approaches demonstrate the effectiveness of visual analytics in revealing and analyzing fairness-related problems. However, there are also limitations to the current approaches. FairVis only supports diagnosing biases in supervised binary classification tasks, and DiscriLens only supports exploring a single sensitive attribute. FairSight explores trade-offs between the group and individual fairness in ranking results, but multigroup fairness remains unexplored. Furthermore, none of these previous systems support model comparison as a mechanism for explaining the impacts of algorithmic debiasing.

2.3 Robust Machine Learning

Robust machine learning is a branch of machine learning research that focuses on developing models that can maintain performance even in the presence of unexpected inputs or perturbations. This is particularly important for applications where the input data is subject to changes, such as autonomous driving or medical diagnosis. One approach to building robust machine learning models is through out-of-distribution (OoD) detection, which involves training a model to identify inputs that are outside the range of its training data. This can help the model avoid making incorrect predictions when presented with inputs that it is not equipped to handle. Another approach is data augmentation, which involves artificially generating additional training data by perturbing existing data points in ways that preserve their original labels. This can help the model learn to be more robust to variations in the input data. Overall, robust machine learning is an important area of research that seeks to ensure that machine learning models can maintain performance in the face of real-world challenges and uncertainties.

2.3.1 Out-of-distribution Detection

Hendrycks et al. (Hendrycks and Gimpel 2018) earlier propose the baseline method for detecting OoD samples by utilizing and distinguishing probabilities from softmax distributions of In-Distribution (ID) and OoD data from neural networks. Based on this, Guo et al. (Guo et al. 2017) leverage temperature scaling to enhance the discrepancy between ID and OoD to make the detection more effective. The following work, Odin, proposed by Liang et al. (Liang, Li, and Srikant 2020) further improves the performance by utilizing both temperature scaling and noise adding. In addition to these softmax-based methods that rely on the model output, some methods represent the confidence of the prediction by modifying the structure of the neural network. For example, Shalev et al. (Shalev, Adi, and Keshet 2019) employ multiple semantic dense representations instead of sparse representations as the target label since they discover the magnitude of the paradigm based on representations is proportional to the confidence of the model prediction. Thus, the learned paradigm of the target label representation is used in this work to determine whether the sample belongs to the OoD sample. Lee et al. (Lee et al. 2017) combine OoD detection with ideas of generative models by suggesting two additional terms added to the original loss, where the first one forces samples from out-of-distribution less confident by the classifier and the second one is for generating most effective training samples for the first one. Similarly, Denouden et al. (Denouden et al. 2018) find that the reconstruction error of variational autoencoder in a generative model can be used for OoD detection since the decoder can efficiently decode the ID samples to generate the reconstructed data corresponding to the Input, while the OoD samples cannot. Thus they utilize reconstruction error and Mahalanobis distance as the metric of OoD detection. In addition, Lakkaraju et al. (Lakkaraju et al. 2016) propose a model-agnostic methodology which is to query an oracle for feedback to both identify unknown unknowns and to intelligently guide the discovery.

Finally, some approaches extract information from the feature of different levels of the model as factors to identify OoD data. Abdelzad et al. (Abdelzad et al. 2019) argue that there may be an implicit Early-Layer Output that can be effectively separated compared to the output layer. The authors extract the input-output data of different layers, use a one-class SVM classifier, count the classification error rate of that layer, and then select the layer with the least error to detect OoD samples. Lakshminarayanan et al. (Lakshminarayanan, Pritzel, and Blundell 2017) leverage Deep Ensembles to predict OoD samples by measuring the OoD degree of them with the help of difference of outputs of classifiers. Lee et al. (Lee et al. 2018) estimate the class conditional Gaussian distributions of different levels of features of the models under Gaussian discriminant analysis, which outputs a Mahalanobis-distance-based confidence score to evaluate the the uncertainty of the given instance. While most prior methods have been evaluated for detecting either OoD or adversarial samples, but not both, this proposed method achieves state-of-the-art performances for both cases in our experiments. In this paper, we employ this method as part of our functionalities that facilitate OoD detection, as it also supports class-incremental learning, which is suitable for serving our defined knowledge validation process.

2.3.2 Data Augmentation

Data augmentation encompasses techniques used to expand and enhance training datasets, addressing the issue of limited training data. For a comprehensive discussion, we refer to the survey by Shorten and Khoshgoftaar (Shorten and Khoshgoftaar 2019). Data augmentation can be achieved through oversampling or data wrapping. Oversampling involves adding synthetic data to the training datasets, while data wrapping focuses on transforming existing data while preserving the original labels. Data-wrapping-based augmentation is further divided into two categories: data manipulation and deep-learning-based augmentation.

Data manipulation involves classical hand-crafted transformations such as flipping, cropping, rotation, and scaling. RGB images can be processed by modifying color channels. Kang et al. (G. Kang et al. 2017) experimented with a kernel filter that randomly swaps pixel values, while Ionue (Inoue 2018) proposed mixing cropped images and assigning labels based on the cropped areas. Inspired by Dropout (Srivastava et al. 2014), Zhong et al. (Zhong et al. 2020) randomly dropped parts of the input image to encourage learning from other regions. Various sophisticated image manipulation methods have been proposed (Jung et al. 2020; Hongyi Zhang et al. 2017; DeVries and Taylor 2017; Hendrycks et al. 2019; Cubuk et al. 2018; X. Chen et al. 2020; Chen, Kornblith, Norouzi, et al. 2020; Chen, Kornblith, Swersky, et al. 2020; Yang et al. 2020; Erichson et al. 2022; Lim et al. 2021; Cubuk et al. 2020).

Deep-learning-based approaches, such as Generative Adversarial Networks (GANs) (Goodfellow et al. 2020; Radford, Metz, and Chintala 2015), manipulate input data by generating new samples, and have been applied to data augmentation (Antoniou, Storkey, and Edwards 2017). Adversarial examples (goodfellow2014explaining), i.e., slightly perturbed versions of original data, are embedded into training data for augmentation in adversarial training (Madry et al. 2017; Hongyang Zhang et al. 2019; Ilyas et al. 2019; Cohen, Rosenfeld, and Kolter 2019). Neural style transfer (Gatys, Ecker, and Bethge 2015) creates new images by combining the style of one image with another's content. Some data manipulation methods adopt an AutoML approach (Cubuk et al. 2018), blurring the boundary between deep-learning-based and data-manipulation-based approaches.

2.3.3 Visual Analytics in Robust Machine Learning

In the field of visual analytics, the work most relevant to robust machine learning is OoD detection and machine learning model behavior diagnosis. OoDAnalyzer (C. Chen et al. 2020) supports context-enriched exploration by designing the grid-based visualization of clustering semantically similar images. The designed saliency maps are provided to explain important regions of the predicted images. Users can perform image comparison through a combination of superposition and juxtaposition, and also explore details of the instances of interest. Similar to OoD data detection, research work emerges for investigating adversarial examples and their corresponding impacts on the machine learning model. AEVis (Cao et al. 2020) is proposed to analyze how adversarial examples fool neural networks. The system takes both normal and adversarial examples as input and extracts their datapaths for model prediction. A multilevel visualization is designed to represent the diverging and merging patterns of the extracted datapaths, which explains how and why the adversarial examples confuse the model through comparing them with normal ones. Ma et al. (ma2019explaining) design a series of visual representations from overview to detail to reveal the progress of poisoned data generated by adversarial attack algorithms that influence the model's prediction.

Many visual analytics techniques are done to understand the model's behavior to improve the model. To date, those techniques can be roughly classified into two categories: structural analysis and instance-level analysis. The structural analysis is performed to deconstruct the network structure and understand the functionalities and contributions neuron by neuron. An earlier approach (Tzeng and Ma 2005) is to directly extract and visualize the graph layouts of neural networks, however, the visualization is not scalable as the structure gets complicated. Liu et al. (M. Liu et al. 2016) further develop CNNVis to visualize deep convolutional neural networks with clustering techniques such that similar neurons and connections are grouped to reduce the visual complexity. Wongsuphasawat et al. (wongsuphasawat2017visualizing) further design an interactive graph-based visualization for exploring the model architecture in Tensorflow (Abadi et al. 2016) by enabling graph transformations from a low-level dataflow chart to a high-level model structure. The system also supports instance-level analysis to explain the relationship between inputs and outputs for improving the model. Rauber et al. (rauber2016visualizing) visualize the representations learned from each layer in the neural network by projecting them onto 2D scatter plots. Through the scatter plots, users can explore clusters from the projections and obtain insights of the representation space learned by the network. Kahng et al. (kahng2017cti) propose ActiVis for revealing the model structure of large-scale deep neural networks through visualizing the computational graph and a projected view that shows the activation relationships between instances. Hohman et al. propose Summit (Hohman et al. 2020), to visualize important neurons and important neuron relationships that contribute to the model prediction. The system integrates an embedding view with an attribute graph view to reveal the activations between classes and to expose influential connections between neurons respectively.

Chapter 3

MACHINE LEARNING SECURITY ANALYSIS

In this chapter, two new frameworks have been introduced that aim to address machine learning vulnerabilities in security aspect. The first framework focuses on adversarial machine learning analysis, which is an emerging area of research that seeks to understand and prevent adversarial attacks on machine learning models. This framework provides visual tools to explore and understand the behavior of these attacks and the vulnerabilities of the models. The second framework is designed for sensitivity auditing of graph mining models, which are commonly used in various applications, such as social network analysis and recommendation systems. This framework enables users to visualize the sensitivity of the models to changes in input data, which is essential for assessing the robustness and reliability of these models. Together, these frameworks represent important advances in the field of visual analytics for machine learning, providing new tools to help users better understand those potential vulnerabilities of the models.

3.1 Vulnerability Diagnosis for Adversarial ML

Recently, researchers have begun identifying design issues and research challenges for defending against adversarial machine learning, such as data de-noising, robust modeling, and defensive validation schemes (Vorobeychik and Kantarcioglu 2018b), citing the need to identify potential vulnerabilities and explore attack strategies to identify threats and impacts. These challenges lend themselves well to a visual analytics paradigm, where training datasets and models can be dynamically explored against the backdrop of adversarial attacks. The preliminary results have developed a visual analytics framework (Figure 2) for explaining model vulnerabilities with respect to adversarial attack algorithms. In general, the framework is designed to answer such questions as: Given the current model, which instances are most vulnerable to be attacked by the given attack algorithms? When the model is compromised by inserting the "poisoned" instances, what is the impact on the model's performance? And how and why do false positives/negatives occur? The framework uses modularized components to allow users to swap out various attack algorithms. A multi-faceted visualization scheme summarizes the attack results from the perspective of the machine learning model and its corresponding training dataset, and coordinated views are designed to help users quickly identify model vulnerabilities and explore potential attack vectors. For an in-depth analysis of specific data instances affected by the attack, a locality-based visualization is designed to reveal neighborhood structure changes due to an adversarial attack.

There are five views that are designed to conduct the analysis: The *Data Table View* summarizes the statistical information for every data instance, and shows the cost in order to attack every instance; the *Model Overview* summarizes the prediction performance for the victim model as well as the poisoned model; the *Data Instances view* presents the labels of the original and poisoned data instances; the *Data Features view*, visualizes the statistical distributions of data along with each feature, and the *Local Impacts view* depicts the relationships between target data instances and their nearest neighbors.

Given the key features of an adversary, we have designed a visual analytics framework that uses existing adversarial attack algorithms as mechanisms for exploring and explaining model vulnerabilities. Our framework is designed to be robust to general adversarial machine learning attacks. However, in order to demonstrate our proposed visual analytics framework, we focus our discussion on *targeted data poisoning attacks* (Biggio and Roli 2018). Data poisoning is an adversarial attack that tries to manipulate the training dataset in order to control

Figure 2. Reliability attack on spam filters. (1) Poisoning instance #40 has the largest impact on the recall value, which is (2) also depicted in the model overview. (3) There is heavy overlap among instances in the two classes as well as the poisoning instances. (4) Instance #40 has been successfully attacked causing a number of innocent instances to have their labels flipped. (5) The flipped instances are very close to the decision boundary. (6) On the feature of words "will" and "email", the variances of poisoning instances are large. (7) A sub-optimal target (instance #80) has less impact on the recall value, but the cost of insertions is 40% lower than that of instance #40.

the prediction behavior of a trained model such that the model will label malicious examples into desired classes (e.g., labeling spam e-mails as safe). Figure 1 (Right) maps the specific goal, knowledge, capabilities, and strategies of a poisoning attack to the generalized adversarial attack.

For the purposes of demonstrating our framework, we assume that the attack takes place in a white-box setting, i.e., the attacker has full knowledge of the training process. Although the scenario seems partial to attackers, it is not unusual for attackers to gain perfect- or near-perfect-knowledge of a model by adopting multi-channel attacks through reverse engineering or intrusion attacks on the model training servers (Biggio, Fumera, and Roli 2014). Furthermore, in the paradigm of proactive defense, it is meaningful to use the worst case attack to explore the upper bounds of model vulnerability (Biggio and Roli 2018). In terms of poisoning operations on the training dataset, we focus on causative attacks (Barreno et al. 2010), where attackers are only allowed to insert specially-crafted data instances. This kind of insertion widely exists in real-world systems, which need to periodically collect new training data, examples include recommender systems and email spam filters (Steinhardt, Koh, and Liang 2017). In such attacks, there is a limit to the number of poisoned instances that can be inserted in each attack iteration, i.e., a budget for an attack. An optimal attack attempts to reach its goal by using the smallest number of insertions within the given budget.

3.1.1 Analytical Tasks

After reviewing various literature on poisoning attacks (Biggio, Nelson, and Laskov 2012; Mozaffari-Kermani et al. 2015; Steinhardt, Koh, and Liang 2017; Shafahi et al. 2018; Suciu et al. 2018; Jagielski et al. 2018; Thomas and Tabrizi 2018; Biggio and Roli 2018), we extracted common high-level tasks for analyzing poisoning attack strategies. These tasks were refined through discussions with our co-author, a domain-expert in adversarial machine learning.

3.1.1.1 T1: Summarize the attack space

A prerequisite for many of the algorithms is to set target instances to be attacked in the training dataset. In our framework, analysts need to be able to identify attack vectors and vulnerabilities of the victim model in order to specify target instances.

3.1.1.2 T2: Summarize the attack results

By following the well-known visual information seeking mantra (Shneiderman 1996), the system should provide a summary of the attack results after an attack is executed. In data poisoning, typical questions that the attackers might ask include:

- *T2.1* How many poisoning data instances are inserted? What is their distribution? Has the attack goal been achieved yet?
- *T2.2* What is the performance of the model before and after the attack and is there a significant difference? How many instances in the training dataset are misclassified by the poisoned model?

3.1.1.3 T3: Diagnose the impact of data poisoning

In this phase, the user explores the prediction results and analyzes the details of the poisoned model. Inspired by the recent work in interpretable machine learning (Alsallakh et al. 2014; Ren et al. 2017; Krause et al. 2017; J. Zhang et al. 2019), we explore the influence of insertion focusing on: attribute changes for individual instances; and drifts of data distributions on features due to poisoning. We consider both instance-level and feature-level diagnoses when investigating the impact of poisoning data. The following questions are explored in this phase:

- *T3.1* At the instance-level, is the original prediction different from the victim model prediction? How close is the data instance to the decision boundary? How do the neighboring instances affect the class label? Is there any poisoned data in the data-instance's top-k nearest neighbors?
- *T3.2* At the feature-level, what is the impact of data poisoning on the feature distributions?

3.1.2 Design Requirements

From the task requirements, we iteratively refined a set of framework design requirements to identify how visual analytics can be used to best to support attack analysis and explanation. We have mapped different analytic tasks to each design requirement.

3.1.2.1 D1: Visualizing the Attack Space

The framework should allow users to upload their victim model and explore vulnerabilities. By examining statistical measures of attack costs and potential impact, the users should be able to find weak points in the victim model depending on the application scenario, and finally identify desired target instances for in-depth analysis in the next step (T1).

3.1.2.2 D2: Visualizing Attack Results

To analyze the results of an attack, the framework should support overview and details-on-demand:

- *Model Overview - D2.1*, summarize prediction performance for the victim model as well as the poisoned model (T2.2);
- *Data Instances - D2.2*, present the labels of the original and poisoned data instances (T2.1, T3.1);
- *Data Features - D2.3*, visualize the statistical distributions of data along each feature (T3.2);
- *Local Impacts - D2.4*, depict the relationships between target data instances and their nearest neighbors (T3.1).

3.1.3 Visual Analytics Framework

Based on the user tasks and design requirements, we have developed a visual analytics framework (Figure 3) for identifying vulnerabilities to adversarial machine learning. The framework supports three main activities: vulnerability analysis, analyzing the attack space, and analyzing attack results. Each activity is supported by a unique set of multiple coordinated views, and the user can freely switch between interfaces and views. All views share the same color mapping in order to establish a consistent visual design. Negative and positive classes are represented by red and blue, respectively, and the dark red and blue colors are used for indicating the labels of poisoning data instances. All actions in our framework are predicated on the user loading their training data and model. While our framework is designed to be modular to an array of attack algorithms, different performance and vulnerability measures are unique to specific attack algorithms. Thus, for discussion and demonstration, we instantiate our framework on data poisoning attacks.

3.1.3.1 Data-Poisoning Attack Algorithms

We focus on the binary classification task described in Figure 4 (a) where the training data instances are denoted as $\boldsymbol{x} \in \mathcal{X}, \mathcal{X} \subseteq \mathbb{R}^{n \times d}$ with class labels of $y \in \{-1, +1\}$ (we refer to the -1 labels as negative and the $+1$ labels as positive). A classification model θ is trained on the victim training dataset, which creates a victim model. For a target data instance $\boldsymbol{x}_t$ and the corresponding predicted label $y_t = \theta(\boldsymbol{x}_t)$, the attacker's goal is to flip the prediction y_t into the desired class $-y_t$ by inserting m poisoning instances $\mathcal{P} = \{\boldsymbol{p}_i | \boldsymbol{p}_i \in \mathbb{R}^d, i \in [1, m]\}$. We use B to represent the budget, which limits the upper bound of m, i.e., an attacker is only allowed to insert at most B poisoned instances. To maximize the impact of data poisoning on the classifier, the attack algorithms craft poisoned instances in the desired class, $y_{p_i} = -y_t$.

Attack Strategies Various attack algorithms have been developed to create an optimal set of $\mathcal{P}$ with $|\mathcal{P}| \leq B$. To demonstrate how attacks can be explored in our proposed framework,

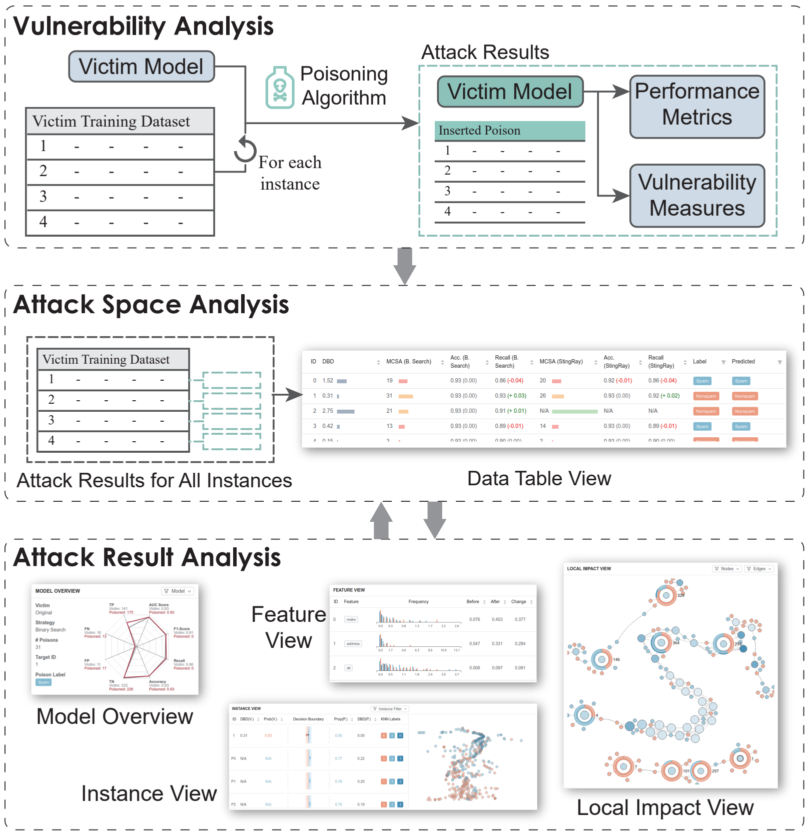


Figure 3. A visual analytics framework for explaining model vulnerabilities to adversarial machine learning attacks. The framework consists of: vulnerability analysis, attack space analysis, and attack result analysis.

we implement two different attack algorithms (Binary-Search and StingRay) described in Figure 4 (b).

Binary-Search Attack[1]. The Binary-Search Attack (Burkard and Lagesse 2017) assumes that the target instance $\boldsymbol{x}_t$ can be considered as an outlier with respect to the training data in the opposite class $\{\boldsymbol{x}_i | y_i = -y_t\}$. The classification model acts as an outlier detector and separates this target from the opposite class $-y_t$. For crafting poisoning instances in a Binary-Search attack, the goal is to establish connections between the target and the desired class $-y_t$

[1]For simplicity, we refer to the Burkard and Lagesse algorithm (Burkard and Lagesse 2017) as "Binary-Search Attack" even though it is not named by the original authors.

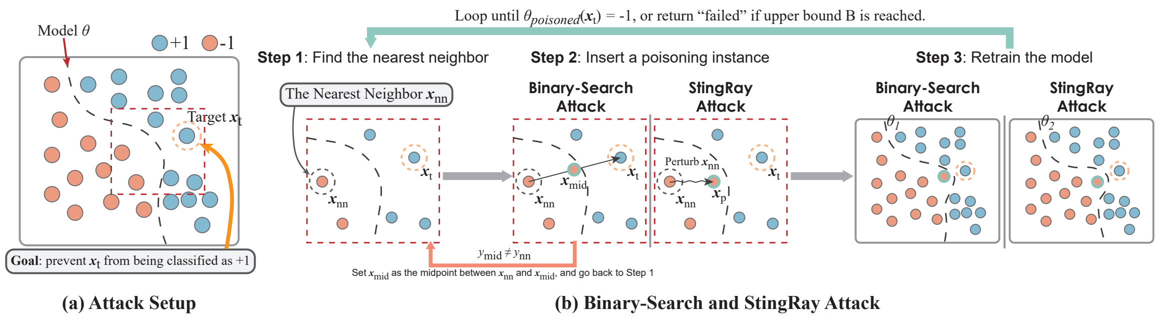

Loop until θpoisoned(xt) = -1, or return "failed" if upper bound B is reached.
Model θ
+1
-1
Target xt
Goal: prevent xt from being classified as +1
Step 1: Find the nearest neighbor
The Nearest Neighbor xnn
xnn
xt
Step 2: Insert a poisoning instance
Binary-Search Attack
StingRay Attack
xmid
Perturb xnn
xp
ymid ≠ ynn
Set xmid as the midpoint between xnn and xmid, and go back to Step 1
Step 3: Retrain the model
θ1
θ2
(a) Attack Setup
(b) Binary-Search and StingRay Attack

Figure 4. An illustration of data poisoning attacks using the Binary-Search and Stingray algorithms. (a) In a binary classification problem, the goal of a data-poisoning attack is to prevent the target instance, $\boldsymbol{x}_t$, from being classified as its original class. (b) The Binary-Search and StingRay attacks consist of three main steps: 1) select the nearest neighbor to the target instance, 2) find a proper poisoning candidate, and 3) retrain the model with the poisoned training data. The procedure repeats until the predicted class label of $\boldsymbol{x}_t$ is flipped, or the budget is reached.

that mitigate the outlyingness of the target. As illustrated in Figure 4, for each iteration, the Binary-Search Attack utilizes the midpoint $\boldsymbol{x}_{mid}$ between $\boldsymbol{x}_t$ and its nearest neighbor $\boldsymbol{x}_{nn}$ in the opposite class, $-y_t$, as a poisoning candidate. If this midpoint is in the desired class, it is considered to be a valid poisoning instance. This instance is appended to the original training dataset, and the model is re-trained (θ_1 in Step 3 - Figure 4). In this way, the poisoned instances are iteratively generated, and the classification boundary is gradually pushed towards the target until the target label is flipped. Sometimes the midpoint may be outside of the desired class. Under this circumstance, a reset of the procedure is required by using the midpoint between $\boldsymbol{x}_{mid}$ and $\boldsymbol{x}_{nn}$ as the new candidate.

StingRay Attack. The StingRay attack (Suciu et al. 2018) inserts new copies of existing data instances by perturbing less-informative features. The StingRay attack shares the same assumptions and pipeline as the Binary-Search attack. The main difference between the attacks is how poisoning instances are generated (Step 2, Figure 4). In StingRay, a base instance, $\boldsymbol{x}_{nn}$, near the target, $\boldsymbol{x}_t$, in the desired class is selected, and a copy of the base instance is created as

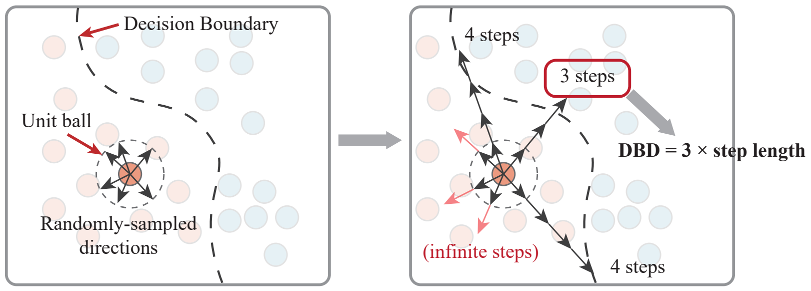


Figure 5. Estimating the decision boundary distance. (Left) Six directional vectors are sampled from the unit ball. (Right) For each direction, the original instance is perturbed one step at a time until it is in the opposite class. In this example, the direction highlighted by the red rectangle is the minimum perturbed step (3 steps) among all the directions.

a poisoned candidate. By using some feature importance measures, a subset of features are selected for value perturbation on the poisoned candidate. After randomly perturbing the feature values, the poisoned instance closest to the target is inserted into the training data.

Attack Results Both attacks insert poisoned data instances into the victim training dataset resulting in the *poisoned training dataset*. The model trained on this poisoned dataset is called the *poisoned model*, and we can explore a variety of performance metrics to help explain the results of an attack (e.g., prediction accuracy, recall). For data instance level analysis (D2.2), we derive two metrics that can characterize the impact of data poisoning on the model predictions.

Decision Boundary Distance (DBD) (He, Li, and Song 2018): In a classifier, the decision boundary distance is defined as the shortest distance from a data instance to the decision boundary. Under the assumption of outlyingness in the Binary-Search or StingRay attack, DBD is an indication of the difficulty of building connections between a target instance and its opposite class. However, it is difficult (and sometimes infeasible) to derive exact values of DBD from the corresponding classifiers, especially in non-linear models. We employ a sample-based, model-independent method to estimate the DBDs for the training data as illustrated in Figure 5. First, with a unit ball centered at the data instance, we uniformly sample a set of unit direction vectors from the ball. For each vector, we perturb the original instance along the vector iteratively with a fixed step length, predict the class label with the classifier, and stop if the prediction is flipped. We use the number of perturbation steps as the distance to the decision boundary. We use the product of step length and the minimum steps among all the directions as an estimation of the DBD for each data instance.

Minimum Cost for a Successful Attack (MCSA): To help users understand the cost of an attack with respect to the budget, we calculate the minimum number of insertions needed to attack a data instance. For each data instance, the MCSA is the number of poisoning instances that must be inserted for a successful attack under an unlimited budget. The MCSA value is dependent on the attack algorithm.

3.1.3.2 Visualizing the Attack Space

The data table view (Figure 2 (B)) acts as an entry point to the attack process. After loading a model, all the training data instances are listed in the table to provide an initial static check of vulnerabilities (*T1*, *D1*). Each row represents a data instance in the training dataset, and columns describe attributes and vulnerability measures which includes the DBD and MCSA for both the Binary-Search and StingRay attack algorithms, as well as the original and the predicted labels. Inspired by Jagielski et al. (Jagielski et al. 2018) and Steinhardt et al. (Steinhardt, Koh, and Liang 2017), we use colored bars for MCSA to highlight different vulnerability levels based on the *poisoning rates*, which is defined as the percentage of poison instances in the entire training dataset. Poisoning rates of lower than 5% are considered to be high risk, since only a small amount of poisoned instances can cause label flipping in these data instances, and poisoning rates of 20% are likely infeasible (high risk of being caught). We define three levels for the poisoning rates: 1) high risk (red) - lower than 5%; 2) intermediate risk (yellow) - 5% to 20%, and; 3) low risk (green) - more than 20% .

The rows in the table can be sorted by assigning a column as the sorting key. The user can click on one of the checkboxes to browse details on the data ID, class label, and feature values, Figure 2 (B). In addition, the clicking operation will trigger a dialog to choose between the two attack algorithms, and the interface for the corresponding attack result will be opened in a new tab page below.

3.1.3.3 Visualizing the Attack Results

After selecting a target instance and an attack algorithm, the user can perform an in-depth analysis of the corresponding attack results. To visualize the results of the attack, we use four views: model overview, instance view, feature view, and kNN graph view.

Model Overview: The model overview provides a summary of the poisoned model as well as a comparison between the original (victim) and poisoned model (*T2*, *D2.1*). The model overview (Figure 2 (C)) provides a brief summary of the names of the victim and the poisoned models, the ID of the target data instance, and the class of the poisoned instances. A radar chart is used to describe the performance of the two models. The four elements commonly used in confusion matrices (true negative (TN), false negative (FN), true positive (TP), and false positive (FP)) are mapped to the four axes on the left side of the radar chart, and accuracy, recall, F1 and ROC-AUC scores are mapped to the right side. When hovering on the lines, the tooltip shows the detailed values on the axes. The two lines in the radar chart can be disabled or enabled by clicking on the legends.

Instance View: The instance view illustrates changes in the training datasets and supports the comparative analysis of predictions made by the victim and poisoned models from the

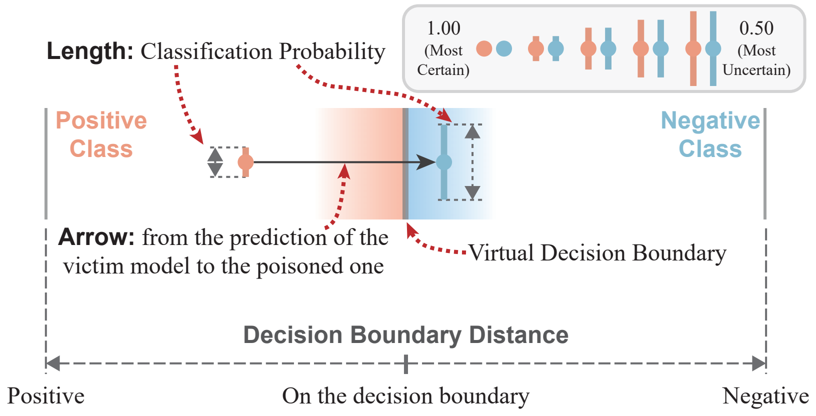

Length: Classification Probability
1.00 (Most Certain)
0.50 (Most Uncertain)
Positive Class
Negative Class
Arrow: from the prediction of the victim model to the poisoned one
Virtual Decision Boundary
Decision Boundary Distance
Positive
On the decision boundary
Negative

Figure 6. Design of the virtual decision boundaries in the instance attribute view. The central vertical line acts as the virtual decision boundary. Two circles representing the prediction results of the victim and the poisoned models are placed beside the line. In this example, the data instance far away from the decision boundary was classified as positive with a relatively high probability. However, in the poisoned model, the instance crosses the boundary, causing the label to flip.

perspective of individual data instances (*T3.1*, *D2.2*). The instance view is comprised of two sub-views, a projection view and an instance attribute view, which visualize data instances under the "overview + detail" scheme.

Projection View: The projection view (Figure 2 (D)) provides a global picture of the data distribution, clusters, and relationships between the original and poisoned instances. We apply the t-SNE projection method (Maaten and Hinton 2008) to the poisoned training dataset. The projection coordinates are then visualized in a scatterplot. We share the colors used in the Model Overview, where red is for label predictions in the negative class and blue for the positive class. To support comparisons between the victim and poisoned model, we apply the corresponding poisoning color to the border of poisoned instances and stripe patterns to the data instances whose class prediction changed after the attack.

Instance Attribute View: The instance attribute view (Figure 2 (E)) uses a table-based layout where each row represents the attributes of an individual data instance including classification probabilities and DBDs from the victim and poisoned model. To conduct a comparison between the attributes of the victim and poisoned models, we embed an illustration of attribute changes into the rows using a virtual decision boundary, Figure 6. Here, the vertical central line acts as a virtual decision boundary and separates the region into two half panes indicating the negative and positive class regions. Two glyphs, representing the predictions of the victim and the poisoned models, are placed in the corresponding half panes based on the predicted class labels. The horizontal distances from the center dots to the central line are proportional to their DBDs. To show the direction of change, we link an arrow from the victim circle to the poisoned circle. Additionally, the classification probabilities are mapped to the length of the lines in the glyph. A set of options are provided in the top right corner of the view for filtering out irrelevant instances based on their types.

Feature View: The feature view is designed to reflect the relationship between class features and prediction outputs to help users understand the effects of data poisoning (*T3.2*, *D2.3*). In Figure 2 (F), each row in the list represents an individual feature. The feature value distribution is visualized as grouped colored bars that correspond to positive, negative, and poisoning data. To facilitate searching for informative features, the rows can be ranked by a feature importance measure on both the victim and the poisoned models. In our framework, we utilize the feature weights exported from classifiers as the measure, e.g., weight vectors for linear classifiers and Gini importance for tree-based models. In the list, the importance values and their rankings from the two models, as well as the difference, are shown in the last three columns.

Local Impact View: In order to understand model vulnerabilities, users need to audit the relationship between poisoned instances and targets to gain insights into the impact of an

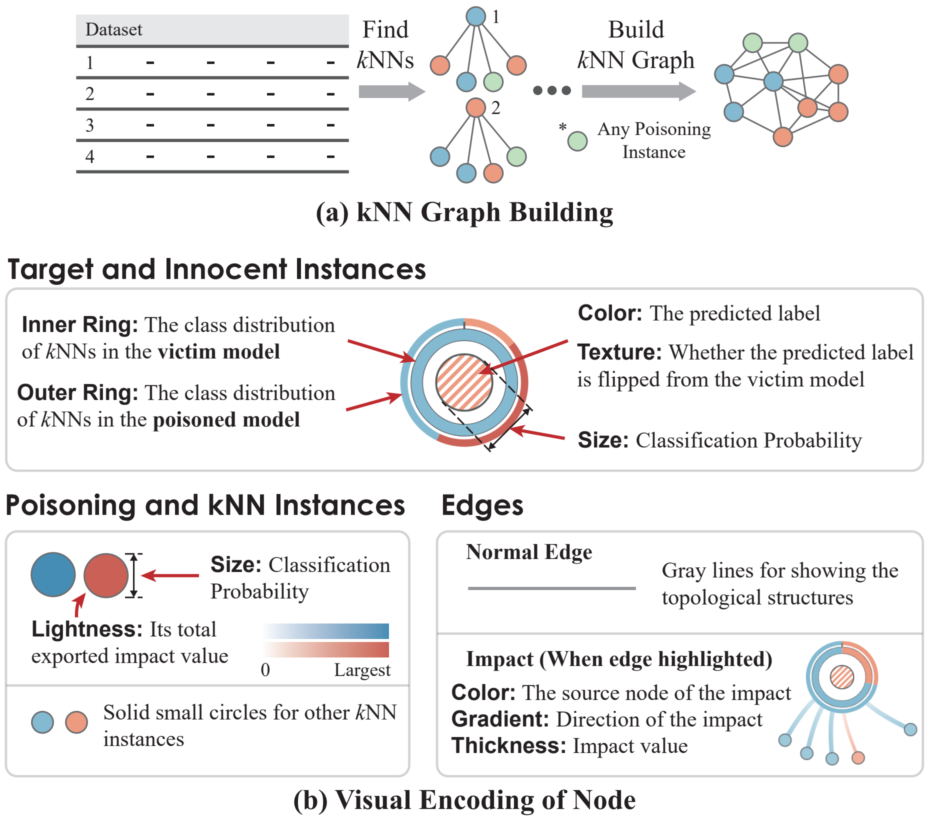

Dataset
1
2
3
4
Find kNNs
1
2
Build kNN Graph
* Any Poisoning Instance
(a) kNN Graph Building
Target and Innocent Instances
Inner Ring: The class distribution of kNNs in the victim model
Outer Ring: The class distribution of kNNs in the poisoned model
Color: The predicted label
Texture: Whether the predicted label is flipped from the victim model
Size: Classification Probability
Poisoning and kNN Instances
Size: Classification Probability
Lightness: Its total exported impact value
0
Largest
Solid small circles for other kNN instances
Edges
Normal Edge
Gray lines for showing the topological structures
Impact (When edge highlighted)
Color: The source node of the impact
Gradient: Direction of the impact
Thickness: Impact value
(b) Visual Encoding of Node

Figure 7. Visual design of the local impact view. (a) The process of building *k*NN graph structures. (b) The visual encodings for nodes and edges.

attack (*T3.1*, *D2.4*). We have designed a local impact view, Figure 2 (G), to assist users in investigating the neighborhood structures of the critical data instances.

For characterizing the neighborhood structures of data instances, we utilize the *k*-nearest-neighbor graph (*k*NN graph), Figure 7 (a), to represent the closeness of neighborhoods, which can reveal the potential impact on the nearby decision boundary. A poisoned instance that is closer to a target may have more impact on the predicted class of the target. Such a representation naturally corresponds to the underlying logic of the attack algorithms, which try to influence the neighborhood structures of target instances. Our view is designed to help the user focus on the most influential instances in an attack. To reduce the analytical burden, we condense the scale of the kNN graph to contain only three types of instances as well as their k-nearest neighbors:

1. The target instance, which is the instance being attacked;
2. The poisoning instances, and;
3. The "innocent" instances, whose labels are flipped after an attack, which is a side-effect of poisoning.

For the target and innocent instances, we extract their kNNs before the attack, i.e., the top-k nearest non-poisoned neighbors. This allows the user to compare the two sets of kNNs to reveal changes in the local structures after inserting poisoned instances.

The design of the local impact view is based on a node-link diagram of the extracted kNN graph where the data instances are represented as nodes. The coordinates of the nodes are computed with the force-directed layout method on the corresponding graph structure. We use three different node glyphs to encode the data instances depending on the instance type (target, poisoned, innocent), Figure 7 (b).

For the target and innocent instances, we utilize a nested design consisting of three layers: a circle, an inner ring, and an outer ring. The circle is filled with a blue or red color representing the predicted label. A striped texture is applied to the filled color if the label predicted by the poisoned model is different from the victim one, indicating that label flipping has occurred for this data instance. Additionally, the classification probability from the poisoned model is mapped to the radius of the circle. The inner ring uses two colored segments to show the distribution of the two classes in the k-nearest non-poisoning neighbors. The outer ring is divided into three segments that correspond to the negative and positive classes and poisoning instances in the kNN.

For poisoned instances, we use circles that are filled with the corresponding poisoning color. To depict the total impact on its neighborhoods, we map the sum of the impact values due to poisoned instances to the lightness of the filled color. As in the encoding of the target instances, the radius of the poisoned instance circles represent the classification probability. All other data instances are drawn as small dots colored by their corresponding prediction labels.

The edges in the local impact view correspond to measures of *relative impacts*, which are represented by directed curved edges. Inspired by the classic leave-one-out cross validation method, the relative impact is a quantitative measure of how the existence of a data instance (poisoned or not) influences the prediction result of another instance with respect to the classification probability. Algorithm 1 is used to calculate the impact of a neighbor $\boldsymbol{x}_{nn}$ on a data instance $\boldsymbol{x}$. First, we train a new model with the same parameter settings as the poisoned model; however, $\boldsymbol{x}_{nn}$ is excluded. Then, we compute the classification probability of $\boldsymbol{x}$ with this new model. Finally, the relative impact value is calculated as the absolute difference between the new probability and the previous one. To indicate the source of the impact, we color an edge using the same color as the impacting data instance. The color gradient maps to the direction of impact and curve thickness maps to the impact value. Additionally, since the kNN graph may not be a fully-connected graph, we employ dashed curves to link the nodes with the minimum distances between two connected components in the kNN graph.

Algorithm 1 Computing the impact of $\boldsymbol{x}_{nn}$ on $\boldsymbol{x}$

```
1: Inputs: training dataset $\mathcal{X}$; two instances $\boldsymbol{x} \in \mathcal{X}$, $\boldsymbol{x}_{nn} \in \mathcal{X}$; previous classification
   probability of $\boldsymbol{x}$, $p_{\boldsymbol{x}}$
2: Outputs: The impact value of $\boldsymbol{x}_{nn}$ on $\boldsymbol{x}$, $I(\boldsymbol{x}_{nn}, \boldsymbol{x})$
3: $\theta \leftarrow$ Classifier$(\mathcal{X} \setminus \{\boldsymbol{x}_{nn}\})$
4: $p'_{\boldsymbol{x}} \leftarrow$ Probability of $\theta(\boldsymbol{x})$
5: $I(\boldsymbol{x}_{nn}, \boldsymbol{x}) \leftarrow |p'_{\boldsymbol{x}} - p_{\boldsymbol{x}}|$ =0
```

The local impact view supports various interactions on the kNN graph. Clicking on a node glyph in the local impact view will highlight the connected edges and nodes and fade out other irrelevant elements. A tooltip will be displayed as well to show the change of neighboring instances before and after the attack. The highlighting effects of data instances are also linked between the projection view and the local impact view. Triggering a highlighting effect in one view will be synchronized in the other one.

One limitation in the proposed design is the potential for visual clutter once the size of the graph becomes considerably large. In order to provide a clear entry point and support detail-on-demand analysis, we support various filters and alternative representations to the visual elements. By default, the edges are replaced by gray lines, which only indicates the linking relationships between nodes. Users can enable the colored curves mentioned above to examine the impacts with a list of switches, Figure 2 (G.1). Unnecessary types of nodes can also be disabled with the filtering options, Figure 2 (G.2).

3.1.4 Case Study

In this section, we present two case studies to demonstrate how our framework can support the analysis of data poisoning attacks from the perspective of models, data instances, features, and local structures. We also summarize feedback from four domain experts.

3.1.4.1 Targeted Attack on Hand-written Digits

Digit recognizers are widely-used in real applications including auto-graders, automatic mail sorting, and bank deposits. In such a system, an attacker may wish to introduce reliability issues that can result in mis-delivered mail, or create targeted attacks that cause checks to be mis-read during electronic deposit. For this case study, we employ a toy example in which a model is used to classify hand-written digits. This case study serves as a mechanism for demonstrating system features.

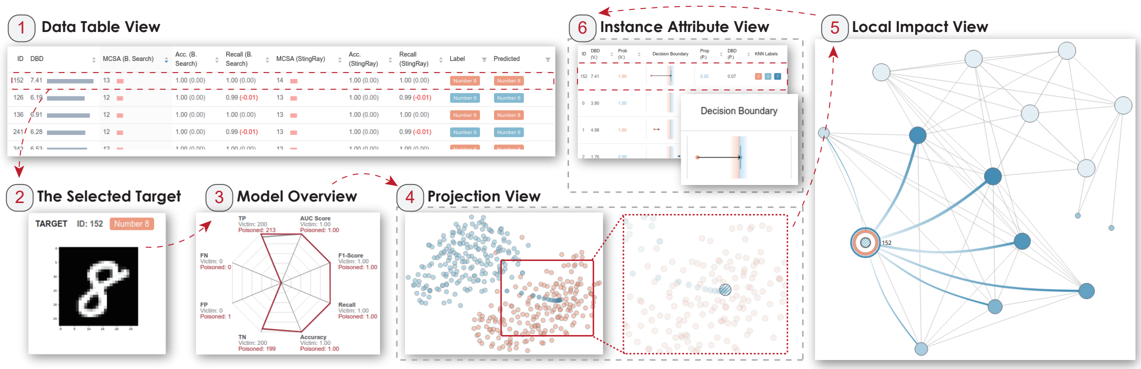


Figure 8. A targeted attack on hand-written digits. (1) In the data table view, we identify the target instance #152 (2) as a potential vulnerability. (3) In the model overview, we observe no significant change of the prediction performance after an attack on #152 occurs. (4) In the projection view, the two classes of instances are clearly separated into two clusters. The poisoning instances (dark blue circles) penetrate class Number 8 and reach the neighboring region of instance #152. (5) The attack can also be explored in the local impact view where poisoning nodes and the target show strong neighboring connections. (6) The detailed prediction results for instance #152 are further inspected in the instance attribute view.

For this classifier, we utilize the MNIST dataset (LeCun et al. 1998), which contains 60,000 images of ten hand-written digits at a 28×28 pixel resolution (784 dimensions in total). We trained a Logistic Regression classifier, implemented in Python Scikit-Learn library (Pedregosa et al. 2011), using 200 randomly sampled images from the numbers 6 and 8, respectively. The value of k for extracting kNN graphs in the local impact view is set to 7.

Initial Vulnerability Check (T1): After the training dataset and model are loaded into the system, vulnerability measures are automatically calculated based on all possible attacks from the Binary-Search and StingRay Attack, and results are displayed in the data table view (Figure 8 (1)). By ranking the two columns of MCSAs for each attack algorithm, the user finds that the red bar colors indicate that many of the data instances are at high risk of a low cost poisoning attack. From the table, the user can also observe that the accuracy and recall values are not highly influenced by an attack, suggesting that a targeted attack on a single instance will not influence prediction performances. To some extent, this may disguise the behavior of a targeted attack by not alerting the model owners with a significant performance reduction.

Visual Analysis of Attack Results (T2, T3): Next, the user wants to explore a potential worst case attack scenario. Here, they select the instance with the largest MCSA among all the data instances (instance #152, 3.5% in poisoning rate) (Figure 8 (2)) under the StingRay attack. As illustrated in Figure 8 (3), first the user performs a general check of the model performance (*T2.2*). In the model overview, the two lines on four performance metrics in the radar chart overlap, indicating little to no model performance impact after a poisoning attack. Next, the user explores the distribution of the poisoning instances (*T2.1, T3.1*). In the projection results, Figure 8 (4), the poisoning instances span the border region of two clusters and flip the prediction of the target instance. However, there are no other innocent instances influenced by the poison insertions. The user can further inspect the impact of at attack on instance #152 by examining the local impact view, Figure 8 (5). Here, the user can observe that in a poisoning attack on instance #152, the neighborhood of #152 must be heavily poisoned, and these poison insertions establish a connection between the target instance #152 and two other blue instances, leading to label flipping. In this case, the user can identify that the sparsity of the data distribution in the feature space may be contributing to the vulnerability of instance #152. Finally, the user explores the detailed prediction result of instance #152 by navigating to the instance attribute view (Figure 8 (6)). Here, the user observes that the label has flipped from Number 8 (red) to Number 6 (blue); however, the poisoning results in a very short DBD and a low classification probability for instance #152.

Lessons Learned and Possible Defense: From the analysis, our domain expert identified several issues in the victim model and dataset. First, even if instance #152 is successfully poisoned, the instance is fairly near the decision boundary of the poisoned model, which can be identified by the low value of DBD and the low classification probability. If any further data manipulations occur in the poisoned dataset, the prediction of the target instance may flip back, i.e., #152 is sensitive to future manipulations and the poisoning may be unstable. For the attackers, additional protection methods that mitigate the sensitivity of previous target instances can be adopted by continuously attacking neighboring instances, further pushing the decision boundary away from the target, or improving attacking algorithms to insert duplicated poisons near the target. Our domain expert was also interested in the pattern of a clear connection from the two blue instances to instance #152 in the local impact view. He noted that it may be due to data sparsity, where no other instances are along the connection path established by the poisoning instances, resulting in #152 having a high vulnerability to poisoning insertions. For defenders who want to alleviate the sparsity issue and improve the security of the victim model, possible solutions could be to add more validated labeled samples into the original training dataset and adopt feature extraction or dimension reduction methods to reduce the number of the original features.

3.1.4.2 Reliability Attack on Spam Filters

For spammers, one of their main goals is to maximize the number of spam emails that reach the customers' inbox. Some models, such as the Naive Bayes spam filter, are extremely vulnerable to data poisoning attacks, as known spammers can exploit the fact that the e-mails they send will be treated as ground truth and used as part of classifier training. Since known spammers will have their mail integrated into the modeling process, they can craft poisoned data instances and try to corrupt the reliability of the filter. These specially-crafted emails can mislead the behavior of the updated spam filter once they are selected in the set of new samples. In this case study, we demonstrate how our framework could be used to explore the vulnerabilities of a spam filter.

We utilize the Spambase dataset (Dua and Graff 2017) that contains emails tagged as non-spam and spam collected from daily business and personal communications. All emails are tokenized and transformed into 57-dimensional vectors containing statistical measures of word frequencies and lengths of sentences. For demonstration purposes, we sub-sampled the dataset into 400 emails, keeping the proportion of non-spam and spam emails (non-spam:spam = 1.538:1) in the original dataset, resulting in 243 non-spam instances and 157 spam ones. A Logistic Regression classifier is trained on the sub-sampled dataset. The value of k for the kNN graphs is again set to 7.

Initial Vulnerability Check (T1): Using the Logistic Regression Classifier as our spam-filter model, we can explore vulnerabilities in the training data. For spam filters, the recall score (True-Positives / True-Positives + False-Negatives) is critical as it represents the proportion of detected spam emails in all the "true" spams. For a spam filter, a lower recall score indicates that fewer true spam emails are detected by the classifier. We want to understand what instances in our training dataset may be the most exploitable. Here, the user can sort the training data instances by the change in recall score after an attack (Figure 2 (1)). After ranking the two columns of recall in ascending order for each attack algorithm, we found that the Binary-Search attack, when performed on instance #40, could result in a 0.09 reduction in the recall score at the cost of inserting 51 poisoned instances.

Visual Analysis of Attack Results (T2, T3): To further understand what an attack on instance # 40 may look like, the user can click on the row of instance #40 and choose "Binary-Search Attack" for a detailed attack visualization. In the model overview, Figure 2 (2), we see that the false negative value representing the undetected spams increased from 16 to 30 (nearly doubling the amount of spam e-mails that would have gotten through the filter), while the number of detected spams decreased from 141 to 127. This result indicates that the performance of correctly labeling spam emails in the poisoned model is worse than the victim model (*T2.2*).

We can further examine the effects of this attack by doing an instance-level inspection using the projection view (*T3.1*). As depicted in Figure 2 (3), the two classes of points, as well as the poisoned instances, show a heavy overlap. This indicates that there is an increased possibility of flipping innocent instances coupled with a decrease in prediction performance. In the local impact view (Figure 2 (4)), it can be observed that the poisoning instances are also strongly connected to each other in their nearest neighbor graph (*T2.1*). Additionally, there are five poisoning instances with a darker color than the others. As the lightness of poisoning nodes reflects their output relative impact, these five neighbors of the target instance contributes more to the prediction results than other poisons. For target instance #40, the outer ring consists only of the poisoning color, indicating that it must be completely surrounded by poisoning instances in order for the attack to be successful. Additionally, in a successful attack, there would be more than 20 innocent instances whose label are flipped from spam to non-spam, which is the main cause of the decreased recall value. After examining the details of these instances in Figure 2 (5), we found that most of their DBDs in the victim model are relatively small, i.e., they are close to the previous decision boundary. As such, their prediction can be influenced by even a small perturbation of the decision boundary. Finally, we conducted a feature-level analysis by browsing the feature view (*T3.2*, Figure 2 (6)). We find that for distributions of poisoning instances along each feature, the variances are quite large on some words including "will" and "email". This suggests that there are large gaps between the non-spam emails and instance #40 on these words in terms of word frequencies, which could be exploited by attackers when designing the contents of the poisoned emails.

Lessons Learned and Possible Defense: From our analysis, our domain expert was able to identify several key issues. First, from the distribution of impact values and classification probabilities among the poisoning instances, an interesting finding was that the poisoning instances close to the target are more uncertain (i.e., of low classification probability values) and essential to flipping its label. Our domain expert mentioned that further optimization may be performed by removing poisoning instances far away from the target because their impact and classification uncertainty could be too low to influence the model training. Second, even though an attack on instance #40 has the maximum influence on the recall value, there is a large (but not unfathomable) cost associated with inserting 51 poisoning instances (poisoning rate = 12.75%). Given the large attack cost, our domain expert was interested in exploring alternative attacks with similar impacts and lower costs, such as instance #80 (Figure 2 (7)). A poisoning attack on #80 can result in a reduction of 0.07 on the recall at almost half the cost of #40 (29 insertions, poisoning rate = 7.25%). The key takeaway that our analyst had was that there are multiple viable attack vectors that could greatly impact the reliability of the spam filter. Given that there are several critical vulnerable targets, the attackers could perform continuous low-cost manipulations to reduce the reliability of the spam filter. This sort of approach is typically referred to as a "boiling-frog attack (Huang et al. 2011)". Here, our domain expert noted that the training-sample selection process may need to be monitored.

3.1.5 Evaluation and Expert Interview

To further assess our framework, we conducted a group interview with our collaborator (E0) and three additional domain experts in adversarial machine learning (denoted as E1, E2, and E3). For the interview, we first introduced the background and goals of our visual analytics framework, followed by an illustration of the functions supported by each view.

Then, we presented a tutorial of the analytical flow with the two case studies described in Section 3.1.4.1 and 3.1.4.2. Finally, the experts were allowed to freely explore the two datasets (MNIST and Spambase) in our system. The interview lasted approximately 1.5 hours.

At the end of the interview session, we collected free-form responses to the following questions:

1. Does the system fulfill the analytical tasks proposed in our work?
2. Does our analytical pipeline match your daily workflow?
3. What are the differences between our visual analytics system and conventional machine learning workflows?
4. Is the core information well-represented in the views?
5. Are there any views that are confusing, or that you feel could have a better design?
6. What results can you find with our system that would be difficult to discover with non-visualization tools?

Framework: The overall workflow of our framework received positive feedback with the experts noting that the system was practical and understandable. E3 appreciated the two-stage (attack space analysis and attack result analysis) design in the interface, and he conducted a combination of "general checks + detailed analysis". E2 noted that *"the stage of attack space analysis gives our domain users a clear sense about the risk of individual samples, so we can start thinking about further actions to make the original learning models more robust and secure, "*. E1 mentioned that the framework could be easily adapted into their daily workflow and improve the efficiency of diagnosing new poisoning attack algorithms. E1 also suggested that it will be more flexible if we can support hot-swapping of attack algorithms to facilitate the diagnosis process.

Visualization: All the experts agreed that the combination of different visualization views can benefit multi-faceted analysis and provide many aspects for scrutinizing the influence of poisoning attacks. E2 was impressed by the instance attribute view and felt that the glyphs were more intuitive than looking at data tables since the changes of distances to the decision boundary can be directly perceived. E3 mentioned that the local impact view provides essential information on how the neighboring structures are being influenced. The two-ring design of the target and innocent instances provides a clear comparison of two groups of nearest neighbors before and after an attack. E3 further added that the node-link diagram and the visual encoding of impacts are effective for tracing the cause of label flipping and the valuable poisoning instances. *"With the depiction of impacts, maybe we could find how to optimize our attack algorithms by further reducing the number of insertions, since some of the low-impact poisoning instances may be removed or aggregated."*

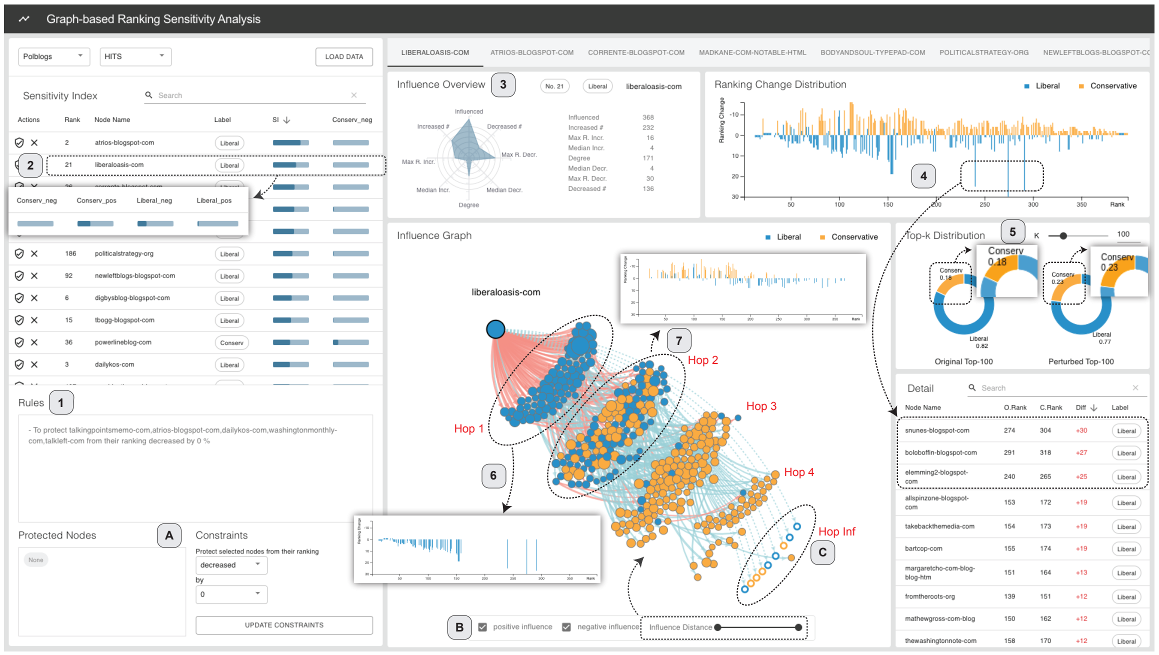
Limitations: One issue found by our collaborator, E0, was the training time that was necessary for using our framework. E0 commented that during the first hour of the interview, we were often required to repeat the visual encoding and functions in the views. However, once the domain experts became familiar with the system after free exploration for some time, they found that the design is useful for gaining insights from attacks. We acknowledge that there could be a long learning curve for domain experts who are novice users in comprehensive visual analytics systems.

3.2 Sensitivity Auditing for Graph Mining

Given the large scale use of graph-based ranking algorithms, we have developed a visual analytics framework to support developers and analysts in exploring and explaining ranking sensitivities. Our framework is designed to be robust to general graph-based ranking algorithms, and supports the removal of nodes as the key perturbation method. While other types of perturbations exist, such as adding/removing nodes/edges(Jia et al. 2020; X. Wang et al. 2018; J. Kang et al. 2018), our framework focuses on the removal of a node and its corresponding edges as a proof-of-concept interaction. The removal perturbation allows us to constrain the computational and exploration space. However, the interactions and design are robust to all types of perturbations, which will be explored in future work. Our potential target audience includes researchers, developers, and analysts who are building and/or deploying graph-based ranking applications. Our goal is to facilitate those experts' analysis of the sensitivity of their chosen ranking methods and support them in auditing the applied ranking algorithms before deployment.

3.2.1 Analytical Tasks

After reviewing recent literature on graph auditing (J. Kang et al. 2018; Kang and Tong 2019), we extracted common high-level tasks for the auditing process. These tasks were refined with our co-authors, domain-experts in graph-mining.



Actions	Rank	Node Name	Label	SI	Conserv_neg
	2	atrios-blogspot-com	Liberal		
	21	liberaloasis-com	Liberal		
	26	corrente-blogspot-com	Liberal		
	186	politicalstrategy-org	Liberal		
	92	newleftblogs-blogspot-com	Liberal		
	6	digbysblog-blogspot-com	Liberal		
	15	tbogg-blogspot-com	Liberal		
	36	powerlineblog-com	Conserv		
	3	dailykos-com	Liberal		

Conserv_neg	Conserv_pos	Liberal_neg	Liberal_pos

- To protect talkingpointsmemo-com,atrios-blogspot-com,dailykos-com,washingtonmonthly-com,talkleft-com from their ranking decreased by 0 %

Influenced	368
Increased #	232
Max R. Incr.	16
Median Incr.	4
Degree	171
Median Decr.	4
Max R. Decr.	30
Decreased #	136

Node Name	O.Rank	C.Rank	Diff	Label
snunes-blogspot-com	274	304	+30	Liberal
boloboffin-blogspot-com	291	318	+27	Liberal
elemming2-blogspot-com	240	265	+25	Liberal
allspinzone-blogspot-com	153	172	+19	Liberal
takebackthemedia-com	154	173	+19	Liberal
bartcop-com	155	174	+19	Liberal
margaretcho-com-blog-blog-htm	151	164	+13	Liberal
fromtheroots-org	139	151	+12	Liberal
mathewgross-com-blog	150	162	+12	Liberal
thewashingtonnote-com	158	170	+12	Liberal

Figure 9. Sensitivity analysis of HITS on political blogs. (1) A predefined rule is set to exclude all perturbations that would cause the rankings of the top-5 blogs to decrease. (2) The blog "liberaloasis.com" has the largest influence under this constraint, and its removal can increase the rankings of the conservative blogs while decreasing the rankings of the liberal blogs. (3) The influence overview indicates that nearly 2/3 of the influenced nodes see a ranking increase. (4) The ranking change distribution view further shows that most of the ranking-increased nodes are conservative blogs and most of the ranking-decreased nodes are liberal blogs, from which the top-3 heavily influenced nodes are ranked 200th or below. (5) The top-*k* proportional view shows that the proportion of liberal blogs decreased from 82% to 77% in the top-100 due to the perturbation.(6, 7) The influence graph view implies that the removal of "liberaloasis.com" has a direct influence on the majority of the liberal nodes (including the top-3 influenced nodes), and as the influence distance increases, more conservative nodes are indirectly influenced.

3.2.1.1 T1: Summarize instance-level sensitivity

The key analytic task is to identify an individual node's ranking and the sensitivity of this node's ranking to changes in the graph structure. Our framework is designed to provide an overview of the ranking results, and enables analysts to explore any node's sensitivity to perturbation. For example:

- *T1.1* Which perturbation causes the largest ranking changes?
- *T1.2* How do perturbations cause ranking changes, i.e., are there topological features that leading to ranking instability?
- *T1.3* Are nodes with specific attributes more sensitive to perturbations in the network, i.e., does the removal of a node of group A lead to changes in the ranking of nodes in group B, where groups are defined by some underlying network attribute.

3.2.1.2 T2: Diagnose the perturbation effects

What-if analysis (Wexler et al. 2019) has previously been used for XAI as a mechanism to investigate machine learning model performance for a range of data features. In our framework, we adopt this idea of what-if analysis to measure the output of any graph-based ranking algorithm by perturbing the input. This enables model developers to measure and explore the ranking changes and corresponding effects. As we focus on *removing nodes* as the perturbation mechanism, a key analytical task is to support diagnosing changes caused by node removals including:

- *T2.1: Summarize the ranking influence of perturbation.* The system should provide a summary of how perturbation has impacted the graph-based ranking results.

- *T2.2: Enable the ranking influence comparison between subgroups.* Each graph node represents an instance and may have attributes/labels that can be used to define class membership. Questions about ranking changes are often strongly tied to questions of fairness related to graph attributes, for example, given a hiring database, are the ranking of female applicants more sensitive to changes in the graph structure than male applicants?
- *T2.3: Identify the topological influence caused by perturbations.* The system also needs to support analysts in exploring how perturbations have influenced the graph topology.

3.2.1.3 T3: Enable progressive analysis

The system should support the analysis of multiple perturbations as analysts explore what-if scenarios.

3.2.2 Design Requirements

From the task requirements, we engaged in an agile design process with our domain experts, iterating over various visualization and interaction designs. Based on our discussions, prototyping and feedback, we have mapped different analytic tasks to a set of design requirements.

3.2.2.1 D1: Visualize the Instance-level Sensitivity

The system should visualize ranking and auditing results for all instances (T1). The view for summarizing the instance-level sensitivity should include the sensitivity index (T1.1) for all nodes with respect to the node attributes (T1.3).

3.2.2.2 D2: Visualize the Effect of Perturbation

The system should be able to guide analysts to explore the perturbation effect of certain node's removal and support interactions such as sorting, searching and filtering to inspect the auditing results and corresponding perturbation effects (T2, T3). This view should include:

- *D2.1 Influence Overview*, which summarizes the perturbation's influence, the degree of ranking changes, and the proportion of nodes whose rankings are increased/decreased, etc. (T1.2, T2.1, T3)
- *D2.2 Distribution View*, which shows how the ranking position changes caused by the perturbation are distributed for each instance and the ranking distribution for each group of nodes. (T2.2, T1.3)
- *D2.3 Ranking Change Detail View*, which lists the influenced nodes for this perturbation. The view should support basic query operations, e.g., sorting, filtering and searching, etc.(T2.1, T3)
- *D2.4 Local Influence Graph View*, which illustrates the relationship between the ranking changes of nodes and the topological changes caused by the perturbation. (T2.3, T3)

The development of visual analytics methods and tools for explainable artificial intelligence (XAI) primarily tackles analytical tasks in vector-space learning, such as classification (Alsallakh et al. 2014; Zhang, Nian Wu, and Zhu 2018), clustering (Kwon, Crnovrsanin, and Ma 2018; Cavallo and Demiralp 2019), and outlier detection (Kwon, Crnovrsanin, and Ma 2018; Xu et al. 2018). However, graph-based learning algorithms are significantly different from vector-space representations, and these differences have not been sufficiently studied in the visual analytics community. Specifically, graph-based ranking algorithms have received little attention; however, algorithms such as PageRank (Page et al. 1999) and HITS (10.1145/324133.324140) are foundational in industrial information retrieval settings. In these information retrieval settings, a person searches a graph-based dataset looking for relevant objects, and the resultant ranking order has a major impact with respect to exposure. For example, recent work by Singh and Joachims (Singh and Joachims 2018) demonstrated that the exposure of resumes to potential employers could be reduced by upwards of 30% if an item's rank fell by as little as three places.

Previous work (Kang and Tong 2019; Ng, Zheng, and Jordan 2001; Chartier et al. 2011; J. Kang et al. 2018) has demonstrated that the results of such graph-ranking algorithms can be highly sensitive to perturbations within the graph structure, and these sensitivity issues give rise to ranking manipulations. Given the importance of the ranking results, it is imperative that algorithm designers and analysts understand the underlying algorithmic sensitivities and vulnerabilities. Consider a news navigation website (Adamic and Glance 2005) where the consumer can search political-related blogs and posts. The search result rankings are determined by a graph ranking algorithm, and higher ranked stories are more likely to be read and shared. Here, one could imagine a nation-state actor that would want to promote biased content. The nation-state actor can create webpages to add various links in the graph structure, or even identify websites to shadow ban, which could manipulate the ranking results so that certain political opinions are more exposed to the public. Given the importance of such rankings, it is critical that model developers have access to tools that can support them in understanding the ranking methods' sensitivity to structural changes in the graph.

The preliminary work has developed a modularized visual analytics framework to facilitate auditing and diagnosing *any* graph-based ranking method's sensitivities by performing what-if analysis over a given graph dataset via node perturbation. This framework is suitable for answering the following questions: Given a list of ranking results performed by a graph-ranking algorithm, how perturbations can influence the results? How such influence is reflected in terms of the topological structures? In this framework, the interactive perturbation of a graph's nodes through coordinated views enables analysts to explore and identify algorithmic sensitivities. A summarization view for the sensitivity index (i.e., the degree of the ranking method's sensitivity to the perturbation) facilitates the identification of the graph-ranking method's instance-level sensitivity. A group of views quantifies the impacts of perturbations through the comparison of statistical information about the ranking results, and a local graph influence view supports the inspection of ranking changes due to changes in the graph topology.

3.2.3 Visual Analytics Framework

Based on the analytic tasks and design requirements, we have developed a visual analytics framework (Figure 10) for auditing, diagnosing, and analyzing graph-based ranking methods' sensitivities to instance-level perturbation through what-if analysis. The framework is designed to first compute a sensitivity calculation for each node of the graph and integrate the results to form a list of all sensitivity information (Figure 10 (A)). Once the precomputation is loaded, the analyst interacts with the system by choosing nodes to perturb, and the framework calculates and visualizes the perturbation effects (Figure 10 (B)). The framework

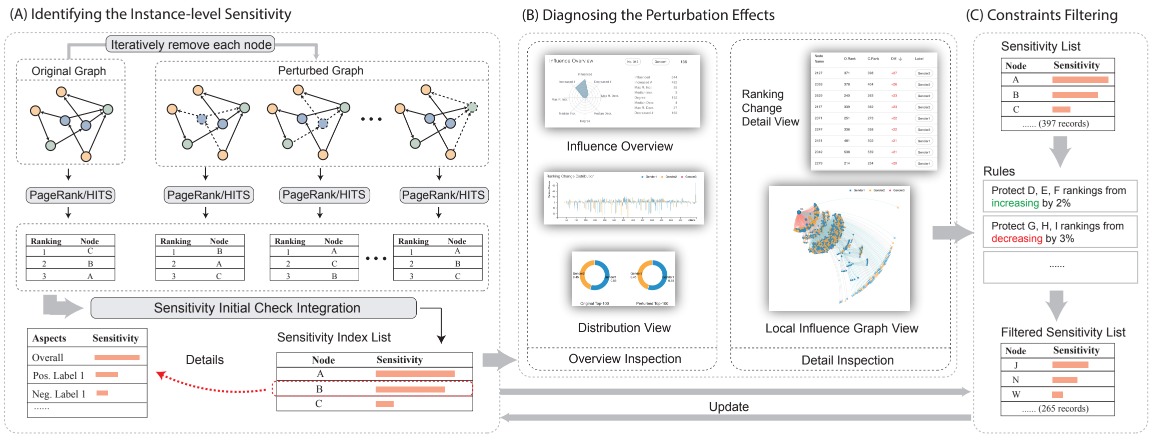

(A) Identifying the Instance-level Sensitivity
Iteratively remove each node
Original Graph
Perturbed Graph
PageRank/HITS
Sensitivity Initial Check Integration
Sensitivity Index List
Details
(B) Diagnosing the Perturbation Effects
Influence Overview
Distribution View
Overview Inspection
Ranking Change Detail View
Local Influence Graph View
Detail Inspection
Update
(C) Constraints Filtering
Sensitivity List
Rules
Protect D, E, F rankings from increasing by 2%
Protect G, H, I rankings from decreasing by 3%
Filtered Sensitivity List

Figure 10. A visual analytics framework for identifying, auditing, and diagnosing a ranking method's sensitivity to instance-level perturbations. The framework consists of Identifying the Instance-level Sensitivity, Diagnosing the Perturbation Effects, and Constraints Filtering. (A) Perturbation is applied for each node instance. The ranking methods are rerun for each new graph generated. Sensitivity to the ranking method for each node's removal is calculated. (B) The analyst can explore one of the nodes to see the perturbation effects through both overview inspection and detailed inspection. (C) Then the analyst can apply constraints to the sensitivity index list based on their findings, and the sensitivity index list is updated.

also supports filtering the list based on the analyst-defined rules to support the inspection of the data under a variety of constraints (Figure 10 (C)). By supporting an iterative process of perturbing nodes and adding analyst-defined rules, this framework enables the auditing of the sensitivity of graph-based ranking algorithms.

The framework supports three main activities: instance-level sensitivity identification, perturbation diagnosis, and customized constraints filtering. Through instance-level sensitivity identification, the analyst can explore an overview of ranking sensitivity with respect to perturbation (node removal). In the perturbation diagnosis, effects caused by the perturbation are displayed with respect to the statistical distribution, top-k distribution, and influenced paths. From the detailed influence view, the analyst can identify the potential constraints and further apply those constraints to filter the results. The analyst can repeat this process until they identify nodes of interest. Our framework is designed to be modular, enabling model designers to integrate any graph-based ranking algorithm. However, for discussion and demonstration purposes, we only explore PageRank and HITS.

3.2.3.1 Identifying the Instance-level Sensitivity

The first component of our framework is designed to support instance-level sensitivity analysis (or auditing) for nodes in the graph. Kang et al. (J. Kang et al. 2018) defined sensitivity auditing as finding the k graph elements (nodes or edges) that have the largest influence on the overall ranking changes of a given graph. This approach identifies the most influential element, removes this element, updates the graph structure, identifies the most influential element from the new graph structure, and continues repeating this process until k elements are found. While such a process is useful for identifying the most sensitive nodes, it does not directly incorporate a mechanism for measuring sensitivity for an individual node. In order to define sensitivity per node, we modify the definition of sensitivity auditing from Kang et al. (J. Kang et al. 2018).

Definition 1. *Given a graph G, an element (a node or an edge) to be removed* el_{rm} *and a graph ranking method* f*, the* graph ranking sensitivity auditing *can be defined as finding the* sensitivity index *for each graph element of G. We denote the sensitivity index of element* el_{rm} *as the degree of ranking method* f*'s sensitivity to the perturbation caused by the removal of this element* el_{rm}*. The sensitivity index of this element* el_{rm} *is represented as:*

$$s[el_{rm}] = sen(f, el_{rm}) = L(rp, rp') \tag{3.1}$$

where rp *and* rp' *denote the ranking positions for each node before and after the perturbation respectively (*el_{rm} *is not included in both* rp *and* rp'*), and* L *stands for a generic difference/distance measure between the ranking vectors before and after perturbation.* s *as the result represents the vector that contains the sensitivity index for every instance, and* $s[el_{rm}]$ *denotes the sensitivity index of* el_{rm}.

Our proposed sensitivity index is used to help analysts compare sensitivities across removals of graph elements. As we focus on node removal in this work, el_{rm} can be replaced by v_{rm}, which denotes the removed node. While there are many metrics available for calculating L, we apply the L_1 norm as the sensitivity metric as it directly measures the accumulated ranking position changes over all nodes. Figure 10 (A) illustrates how the Instance-level sensitivity module performs the initial sensitivity index check on each instance. Applying Definition 1, our framework first calculates the ranking change for every node by removing each node (and its corresponding edges) from the graph and calculates the ranking method on the perturbed graph. The removal of a node may cause the ranking of other nodes to change in both ranking directions (increase or decrease). As such, the sensitivity can be summarized in multiple ways. For example, we could compute the overall positive/negative ranking change (or influence) that occurs when removing a node could be summarized or the class-specific positive/negative influence, where classes of nodes are defined based on their attributes and labels. In our framework, we calculate both a positive sensitivity index sen_p and a negative sensitivity index sen_n with respect to class labels as follows:

$$s^b_{pos}[v_{rm}] = sen_p(f, v_{rm}, b) = \begin{cases} \sum |rp_v - rp'_v|, rp_v - rp'_v > 0 \\ 0, otherwise \end{cases} \tag{3.2}$$

$$s^b_{neg}[v_{rm}] = sen_n(f, v_{rm}, b) = \begin{cases} \sum |rp_v - rp'_v|, rp_v - rp'_v < 0 \\ 0, otherwise \end{cases} \tag{3.3}$$

where $v \in V \wedge v \neq v_{rm} \wedge label(v) = b$, V is the set of all nodes in the graph, b is the class label, and rp_v and rp'_v are the ranking positions of node v before and after the perturbation respectively. $\mathbf{s_{pos}}$ and $\mathbf{s_{neg}}$ represent the vectors of positive and negative sensitive index for each instance, in which $s_{pos}[v_{rm}] = \sum_{b \in B} s^b_{pos}[v_{rm}]$ and $s_{neg}[v_{rm}] = \sum_{b \in B} s^b_{neg}[v_{rm}]$, where B is a set of labels. Eq. 3.1, Eq. 3.2 and Eq. 3.3, are applied in Algorithm 2 to realize the the initial sensitivity index check for every node and every available label. The system then visualizes the output in the sortable sensitivity index list (D1), which shows each node's current ranking and sensitivity indices with respect to the node's class label(s).

3.2.3.2 Diagnosing the Perturbation Effects

The second component of our framework is designed to support what-if analysis. In this module, analysts begin by selecting a node from the sensitivity index list to further explore the perturbation effects. Several linked views are deployed:

Influence Overview: The influence overview provides basic information on changes caused by removing a specific node (D2.1). These changes include 1) the number of influenced nodes which have ranking changes after the perturbation; 2, 3) the number of influenced nodes whose ranking increased/decreased after the perturbation; 4, 5) the max/min of increased/decreased ranking changes; 6, 7) the median of increased/decreased ranking changes; and 8) the degrees of the node. These 8 metrics provide the analyst with a statistical overview of the ranking influence of nodes due to perturbations. The radar chart is used to provide an overview of the sensitivity metrics with respect to the effects of a perturbation. (Figure 9 (3)) The radar chart allows for the further addition of new metrics and can also preserve the overall information of the perturbation when the analyst switches between multiple perturbation diagnoses through the tabs on the top of the view.

Influence Distribution View (D2.2): In addition to showing the snapshot of ranking

Algorithm 2 Sensitivity Index Initial Check

```
1:  *Inputs:* graph data $G$; ranking method $f$; node labels B;
2:  *Outputs:* overall SI vector *s*; positive/negative SI vector $\mathbf{s_{pos}}/\mathbf{s_{neg}}$; positive/negative SI
    vector in terms of labels $\mathbf{S_{pos}}/\mathbf{S_{neg}}$;
3:  $\mathbf{r_{original}} \leftarrow f(G)$
3:  for each node $v$ in $G$.nodes do
4:  remove $v$ and all connected edges *e* from $G$
5:  $\mathbf{r_{removed}} \leftarrow f(G)$
6:  calculate $\mathbf{r_{diff}}$ with $\mathbf{r_{original}}$ for each node in $\mathbf{r_{removed}}$
6:      for each $r_{diff}[i]$ in $\mathbf{r_{diff}}$ do
6:          if $r_{diff}[i] > 0$ then
7:  $s^b_{pos}[v] \leftarrow s^b_{pos}[v] + abs(r_{diff}[i])$ for $b$ = label(i)
7:          end if
7:          if $r_{diff}[i] < 0$ then
8:  $s^b_{neg}[v] \leftarrow s^b_{neg}[v] + abs(r_{diff}[i])$ for $b$ = label(i)
8:          end if
9:  $s[v] \leftarrow s[v] + abs(r_{diff}[i])$
9:      end for
10: $s_{pos}[v] \leftarrow sum(s^b_{pos}[v])$; $s_{neg}[v] \leftarrow sum(s^b_{neg}[v])$ for $b$ in B
11: add all $s^b_{pos}[v]$ to $S_{pos}[v]$ and all $s^b_{neg}[v]$ to $S_{neg}[v]$ for $b$ in B
12: add $v$ and *e* back to $G$
12: end for
13: *Return* $\mathbf{s}$, $\mathbf{s_{pos}}$, $\mathbf{s_{neg}}$, $\mathbf{S_{pos}}$, $\mathbf{S_{neg}}$ =0
```

changes, we also provide a ranking change distribution view and the top-k proportional distribution view.

Ranking Change Distribution View: A bar chart (Figure 11) is used to show the ranking change distribution. Each bar is a node. The position of the bar on the x axis denotes the original ranking position for the node. The height of the bar on the y axis denotes the ranking change for the node. Colors represent the node labels. We scale the axes of the bar chart such that a 90-degree clockwise rotation of the bar also allows the analyst to infer the future rank of the node.

Top-k Proportional Distribution View: Chartier et al. (Chartier et al. 2011) noted that when applying graph-based ranking algorithms for search engines, there could be an argument that ranking changes of webpages that are not part of the top-k ranking are less important than those in the top-k. This argument can be extended to any general graph ranking problem, where the analyst can choose a k for which elements below this ranking will not be considered. For example, a hiring manager may not be interested in resumes ranked outside of the top-25, but it important to understand whether certain node attributes are underrepresented in the top-k. In the Top-k Proportional Distribution View, we use two donut charts to represent the proportions of nodes of different categories belonging to ranking 1 to ranking k before and after the perturbation (Figure 9 (5)), and k is interactively specified.

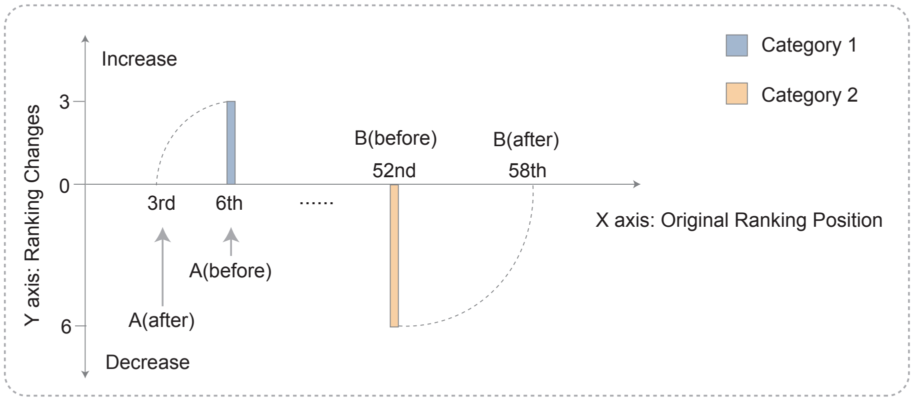


Figure 11. Visual encoding of the Ranking Change Distribution View. We use a bar chart to show the ranking change distribution. Each bar is a node. The position of the bar on the x axis denotes the original ranking position for the node. The height of the bar on the y axis denotes the ranking change for the node. Colors represent the node labels. We scale the axes of the bar chart such that a 90-degree clockwise rotation of the bar also allows the analyst to infer the future rank of the node.

Influence Detail View: In the influence detail view, a data table is used to give the exact details about the node name, previous ranking, perturbed ranking, ranking difference, and labels. The analyst can sort all columns in the table, and, by hovering over a row, the location of the corresponding node will be highlighted in the local influence graph.

Influence Graph View: While summarizing the changes in rank is important, our domain experts also required the ability to explore the impacts on the graph topology caused by perturbations. Note that for the graph-based ranking algorithm, such as PageRank and HITS, the underlying logic propagates the ranking value until convergence. In other words, the probability value for each node is propagated via a directed link connected between a node and its predecessors [2]. In our case, perturbation is equivalent to removing the designated node and the links associated with it. As such, perturbation may cause two types of influence: *direct influence* and *indirect influence*. We consider direct influence to be the ranking changes of nodes due to the perturbation of their predecessors. Indirect influence is the ranking changes of nodes due to perturbations of any nodes that are not their predecessors. Understanding the direct and indirect influence is critical as analysts are interested in nodes that cause limited direct influence but have larger amounts of indirect influence. These types of nodes are prime candidates for shadowing banning and other types of attacks as their removal can greatly influence the ranking results, but the removal will be relatively unnoticed by their direct connections, making such an attack difficult to quickly identify.

To perform the direct/indirect influence analysis, we begin by building the influence graph which contains the topological relationships between the influenced nodes and the removed node. Here we introduce the concept of *influence distance*, which we define as the geodesic distance [3] between two nodes. We denote the influenced nodes, which are successors [4] of the removed node, as *hop-1* influenced nodes, which are also represented as the directly influenced nodes. Hop-1 influenced nodes have an influence distance of 1. Similarly, from the influenced nodes, we denote the hop-1 influenced nodes' successors as hop-2 in-

[2]A node that has a link that points to a given node in a directed path.

[3]The length of the shortest directed path connecting the two nodes. For an unweighted graph, the length is the number of edges in the shortest path. The distance is *infinite* if there is no path between two nodes.

[4]Any node whose geodesic distance is equal to one from a path starting at i.

fluenced nodes, which have an influence distance 2, and so on and so forth. We define the influence distance to be the shortest distance between the removed node and the influenced node, which means that a hop-1 influenced node may also be a hop-3 influenced node's successor, but we only consider it as hop-1. Finally, for those nodes whose influence distance is infinite, we denote them as hop-inf influenced nodes. Algorithm 3 details our influence graph construction algorithm.

Algorithm 3 Influence Graph Construction

```
 1: *Inputs:* graph data $G$; removed node $v_r$; influenced nodes $V'$
 2: *Outputs:* influence graph $G'$
 3: initialize $G'$
 4: initialize queue $\mathbf{q}$
 5: $G'$.addNode($v_r$, hop=0)
 6: $\mathbf{q}$.push($v_r$)
 6: while $\mathbf{q}$ is not empty do
 7: $v \leftarrow \mathbf{q}$.pop()
 7:     for $v_{neighbor}$ in $v$.neightbors() do
 7:         if $v_{neighbor}$ is influenced and not visited then
 8: visited($v_{neighbor}$) $\leftarrow$ true
 9: $G'$.addNode($v_{neighbor}$, hop=$v$.hop + 1)
10: $G'$.addEdge($v$, $v_{neighbor}$)
11: $\mathbf{q}$.push($v_{neighbor}$)
11:         end if
11:     end for
11: end while
11: if any influenced node not in $G'$ then
11:     for $v_{remain}$ in remained influenced nodes do
12: $G'$.addNode($v_{remain}$, hop=inf)
13: $G'$.addEdge($v$, $v_{remain}$)
13:     end for
13: end if
14: *Return* $G'$ =0
```

We visualize the influence caused by removing/perturbing a node (D2.4) as a customized radial graph layout (Figure 12). In this customized layout, the removed node is set as the

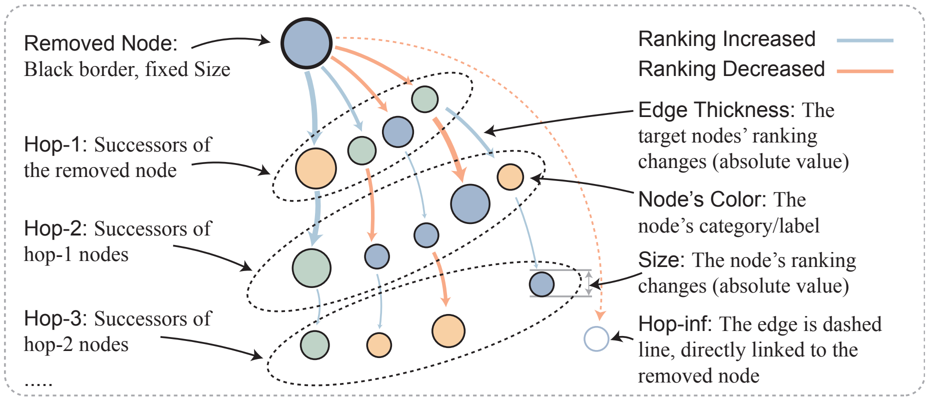

Removed Node: Black border, fixed Size
Hop-1: Successors of the removed node
Hop-2: Successors of hop-1 nodes
Hop-3: Successors of hop-2 nodes
.....
Ranking Increased
Ranking Decreased
Edge Thickness: The target nodes' ranking changes (absolute value)
Node's Color: The node's category/label
Size: The node's ranking changes (absolute value)
Hop-inf: The edge is dashed line, directly linked to the removed node

Figure 12. Visual encoding of the Influence Graph View. The perturbed node is placed at the left top of the graph and all successors which have ranking changes are grouped as hop-1 nodes. The successors of any hop-1 nodes which have ranking changes are grouped as hop-2 nodes, so on and so forth. The nodes' colors encode their categories (using colored filling and black strokes for regular nodes, and white filling and colored strokes for hop-inf nodes), and the nodes' sizes and thickness of their incoming edges encode the absolute value of ranking changes of nodes. The orange and light blue edge colors denote a ranking decrease/increase respectively. Nodes that do not have connections with any of the nodes in the influence graph are hop-inf nodes and are connected to the perturbed node with a dashed arrow line.

center of the force, and the strength of the charge force for each type of the node (hop-1 node, hop-2 node, etc.) is increased gradually based on the number n of hop-n. In this way, all the nodes are clustered, and the influence graph forms a tree-like structure where the root of the tree starts from the top-left of the view and the branches spread towards the bottom-right of the view. Compared with a traditional force directed layout, this layout has two advantages: 1) All the influenced nodes are organized and positions of the nodes are relatively fixed in this layout, which enables the analyst to preserve their mental model. 2) Nodes of the same type are clustered in this view, enabling the analyst to explore the composition for each cluster, i.e., each group of hop-n nodes. The color of the edges is encoded with light blue and orange, which shows whether the influenced node increased/decreased. The nodes' colors encode their categories (using colored filling and black stroke for regular nodes, and white filling and colored stroke for hop-inf nodes), and the nodes' sizes and thickness of their incoming edges encode the absolute value of the ranking changes of nodes. We also provide interactive graph filtering so that analysts can filter by nodes whose rankings have increased/decreased, or nodes within certain influence distance ranges. The ranking change distribution view will also automatically update based on the ranges, Figure 9 (6) (7). By filtering, the analysts can answer questions related to the perturbation, such as whether the node has a large direct influence on specific categories, or whether the node has a large indirect influence on nodes that are far away.

3.2.3.3 Customized Constraints Filtering

As we hinted at with the discussion of exploring ranking changes with respect to node attributes, in real-world applications, measures of sensitivity may need to be done with respect to domain specific constraints. Consider the Google search engine as an example, where each website is more concerned about reaching the first page of search results as well as climbing the ranking on that first page. It is reported that Google traffic is captured by 91.5% of the first page search results (*How Valuable Is The First Page of Google?*), and there are also reports suggesting that top-3 results capture upwards of 75% of the clicks (*We Analyzed 5 Million Google Search Results. Here's What We Learned About Organic CTR*). In such a setting, a domain owner would be interested in how sensitive their website is to ranking changes as a change in ranking from the first page of the search results to the second can have disastrous implications for web traffic.

Our framework enables customized constraints filtering functionality that allows analysts to add constraints to the sensitivity index list. Specifically, the analyst can define constraints that prevent selected nodes' rankings from increasing/decreasing by a certain degree. As Figure 10 (C) shows, the analyst may have domain knowledge of the data and may wish to

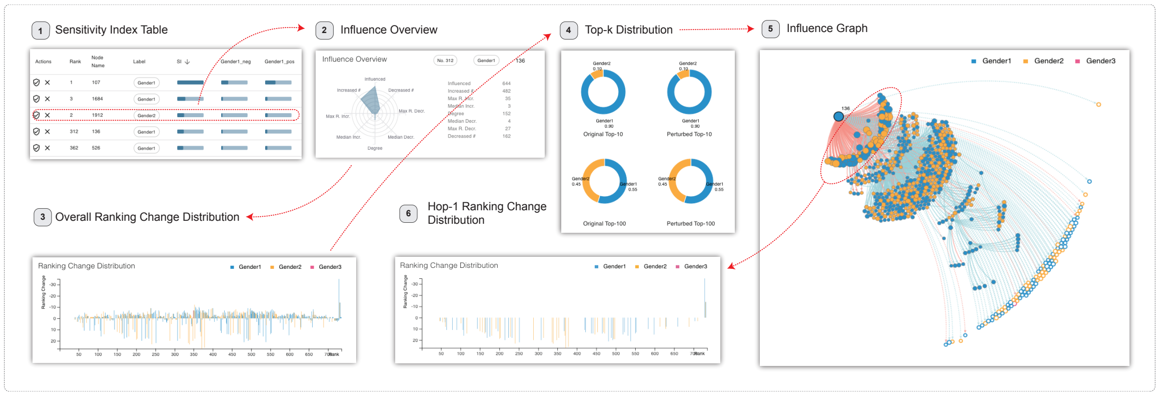


Figure 13. Facebook sensitivity analysis on PageRank. (1) The sensitivity index list shows that *user 136* has the 4th largest influence on the Sensitivity Index while its ranking score is 312 out of 734 nodes, which is considerably low for such a large influence. (2) The influence overview shows that the removal of *user 136* influences 644 of the 734 nodes, with 482 of them being positively influenced (ranking increased) while 162 are negatively influenced (ranking decreased). (3) The ranking change distribution view shows that negatively influenced nodes see a larger ranking decrease in comparison to positively influence nodes. (4) The top-k distribution view shows that as k increases from 10 to 100, the proportion of gender 2 increases, which means nodes of gender 1 are ranked relatively higher than nodes of gender 2. (5) The influence graph view further explains that most of those nodes who have large ranking changes are the neighbors of the removed *user 136*, and (6) the corresponding distribution view shows that those neighbors' rankings are evenly distributed.

add constraints to the sensitivity index list to filter out any perturbations that would violate the constraints. For example, if the analyst wants all the possible perturbations on the sensitivity index list that do not cause the top-3 nodes to experience ranking drops, the analyst can add a constraint: prevent top-3 nodes from ranking decreased by 0. The analyst can then sort the sensitivity index list and add the top-3 ranked nodes to the protected list by clicking the shield-like button for each of them and then configure a new rule "protect selected nodes from ranking decreased by 0". Finally, the analyst clicks the *Update Constraints* button to add the new rule. The newly configured rule will then be displayed on the *Rules* section (Figure 9 (1)) and the sensitivity index list will be automatically updated such that any potential perturbations in it will not result in the top-3 ranked nodes having their rank decreased. The analyst can also add more constraints as they may find certain nodes need to be protected from the perturbation during the diagnosis process. For example, if the analyst finds that a perturbation causes a significant ranking drop on the node, the analyst can use the lasso tool to select the node in the influence graph view (the node with significant ranking changes is encoded as a large circle in this view), and add it to the protected nodes and then apply a new rule. The analyst can then restart exploring the potential perturbations that do not violate the new rule. In this way, the analyst is able to explore possible perturbations under a variety of customized constraints.

3.2.4 Case Study

In this section, we present three case studies to demonstrate how our framework supports sensitivity auditing for graph-based ranking. We showcase how data scientists analyze the sensitivity of the PageRank algorithm on Facebook social network data, how ranking developers check the robustness of their graph-based ranking algorithms, how social scientists analyze the exposure of blogs, and how the sensitivity of ranking algorithms can help identify potential manipulations.

3.2.4.1 Facebook Ranking with PageRank

In social network analysis, ranking members based on the graph structure is essential to tasks such as advertising (Heidemann, Klier, and Probst 2010), social link prediction (gleich2015pagerank), and recommendation (Gori:2007:IRB:1625275.1625720). Perturbations in rankings may have a significant influence on the related business strategies. As such, it is important to audit the sensitivity of such rankings as this may help uncover malicious accounts, or reveal unintended biases in the algorithm. For example, analysts employ graph-based ranking methods to provide recommendations within a social network. These recommendations are based on the ranking results which predict who in the social network might be interested in the recommended products. Here, it may be important to understand if one sub-population is being over-targeted with particular advertisements. In this case study, we analyze ranking sensitivity in the Facebook social network dataset (Leskovec and Mcauley 2012). For demonstration purposes, the social network is down-sampled into a graph with 734 nodes and 74254 edges. We use the gender of each network member (where each member is a node in the graph) as the class label and explore if perturbations in the network reveal ranking bias with respect to gender.

Identifying the Instance-level Sensitivity (T1): The sensitivity index table is displayed after loading the graph data and choosing the ranking method, Figure 13. We sort the column *SI (Sensitivity Index)* in order to explore which node (network member) is likely to have the largest influence on the ranking result. After sorting by SI, it can be observed that the nodes with the highest SI are also the highest ranked nodes. This phenomenon matches the explanation in Kang et al. (J. Kang et al. 2018) that the nodes with high influence are also often highly ranked. Interestingly, though, we see that the 4th node, *user 136*, has a high sensitivity index. However, the network member is ranked 312th out of 734 nodes. This particular case is of keen interest to our analyst.

Diagnosing the Perturbation Effects (T2): By clicking 'diagnose the perturbation effect' on *user 136*, the details of the influence is depicted in the influence overview. In Figure 13 (2), the influence overview shows that the perturbation caused by removing *user 136* has influenced 644 out of 734 nodes, where 482 nodes' rankings increased and 162 nodes' rankings decreased. By further exploring the ranking change distribution view, we find that although positively influenced nodes are 3 times more common than the negatively influenced nodes, the negatively influenced nodes are subject to larger ranking fluctuations than the positively influenced nodes. While we expect that the nodes that were previously ranked lower than *user 136* would see a rise in ranking to fill the gap created by *user 136*, it is surprising to also find large negative changes occurring. In the ranking distribution view and the top-k distribution view, Figure 13 (3), we observe that the perturbation does not result in drastic changes in the ranking distributions. Furthermore, if we split the top-k rankings by gender, the perturbation is not observed to impact one gender class's rankings more than another.

In addition, our analyst is interested in the relationships between the ranking changes and the topological structure. Specifically, the analyst wonders how the removed *user 136* is connected to the influenced nodes given the stronger impacts for decreased ranking. In the local influence graph view, the nodes who have significant ranking changes in the perturbation are directly connected to *user 136* since all the large circles are hop-1 node, Figure 13 (4). After selecting the range of influence distances, we can see that the decreases of rankings for most of the nodes are caused by being the immediate successor of *user 136*. Furthermore, most of the nodes outside the hop-1 circle have their ranks slightly increased. This may be due to the fact that those nodes are not heavily influenced by the perturbation.

3.2.4.2 Subreddit Ranking with PageRank

In the second case study, we audit the sensitivity of PageRank results on the subreddit community interaction graph dataset (Page et al. 1999). Kumar et al. (Kumar et al. 2018) studied the community interactions and conflicts between communities in Reddit to show that a community can be mobilized by negative sentiment comments from another community. Such conflicts between communities can potentially reduce the activities among community members and may lead to people leaving the platform. In such cases, it is also possible that other communities will be influenced due to chain effects in the network, which may cause

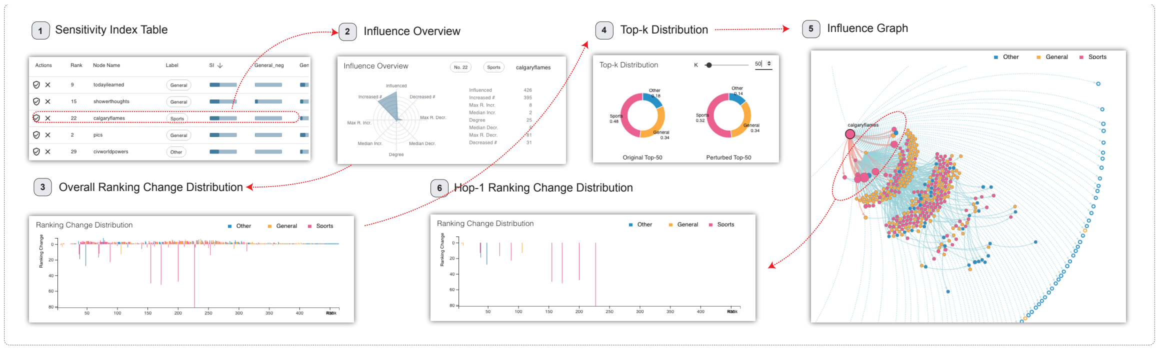

1 Sensitivity Index Table
2 Influence Overview
3 Overall Ranking Change Distribution
4 Top-k Distribution
5 Influence Graph
6 Hop-1 Ranking Change Distribution
Ranking Change Distribution
Other
General
Sports
Original Top-50
Perturbed Top-50
calgaryflames

Figure 14. Subreddit sensitivity analysis on PageRank. (1) The sensitivity index list shows that the "r/CalgaryFlames" node belongs to the sports topics and has a large sensitivity index. (2) The influence view shows that removal influenced 426 out of 464 nodes, and 395 of them are positively influenced while only 31 are negatively influenced. (3) The ranking change distribution view shows that a few nodes experience a large ranking decrease due to the removal of "r/CalgaryFlames", the majority of which are sports topics. (4) The top-k distribution view shows that the subreddits for sports occupy more of the top-50 ranks than other subreddit topics. (5) (6) The removed node has relatively few neighbors that are negatively influenced, and there are more direct influences in the 'General' nodes than 'Other' nodes.

member churn and increased complaints about the platform. Thus, the Reddit community managers and data analysts may be interested in inspecting the activities of communities to make sure the content environment is benign and also be aware of any perturbations (deletion of subreddit posts/activity reduction) that might influence ranking results during any possible recommendation processes.

In this case, we sub-sampled the subreddit community interaction dataset (Kumar et al. 2018) down to 464 nodes and 6676 edges. Each node represents a subreddit, and a topic label is assigned to the nodes to identify their main categories, such as "r/basketball", "r/soccer" and "r/nfl" under the topic "Sport". An edge between two nodes indicates there is a comment in one subreddit which referred to the other subreddit in the content of the comment. We select three topics to explore: Sports, General, and Others. In this case, comments with a high page rank would receive more views, and if negative comments generate more clicks, this could be of concern. Community administrators could utilize our framework to explore the interactions between communities and identify potential issues of flaming, karma farming, etc.

Identifying the Instance-level Sensitivity (T1): After the ranking results are loaded into the system, we want to know which subreddit in the "Sports" topic has the largest influence on the reputation of other subreddits. By sorting the sensitivity index column, the ranking result is listed in the sensitivity index table, and the subreddits are ordered by their sensitivity values in descending order. We believe that concrete entities such as sports players and teams can have more controversial topics, but are less noticeable by community members who are not interested in them. Thus, we skip the general subreddits such as "r/hockey", "r/baseball", and "r/basketball" and focus on the "r/CalgaryFlames", which has a relatively large influence on the reputation rankings of the subreddits.

Diagnosing the Perturbation Effects (T2): The impact of removing the "r/CalgaryFlames" node are shown on the right side of Figure 14. In the influence overview, the removal has influenced 426 out of 464 nodes. Among the 426 nodes, 395 of them have their ranks increased while 31 decreased. Specifically, the node "r/thebeach" has increased by 8 positions, and the rank of node "r/coloradoavalanche" has dropped by 81 positions. This indicates that the perturbation has triggered massive declines even though the average increase is relatively low. In the ranking change distribution view, we observe that the ranking declines occur primarily in the Sports subreddits whose original ranks are relatively low (around 255 out of 464). Compared with the decrease, the overall distribution of the increased nodes covers a broader range on the original rankings; however, a much smaller climbing effect in the ranking positions is observed. This may be due to the fact that the original rank of the "r/CalgaryFlames" is very high (22nd), which could possibly bump many lower-ranked subreddits into higher positions. In the distribution view, all three topics (sports, general and others) receive slight increases in the median values. However, the overall distribution remains the same. We further query the top-50 since there are considerable ranking changes depicted in the ranking change distribution view. This suggests that after removing the "r/CalgaryFlames", the proportion of subreddits in the category of Sports among the top-50 has increased from 48% to 52%, while the proportion of "Other" subreddits has dropped from 18% to 14%. In the influence graph view, we find that the removal results in large drops the to ranks of the hop-1 nodes, which matches the patterns shown in the ranking change distribution view. That is to say, the removal of "r/CalgaryFlames" only influences its neighbors with significant ranking changes. By filtering on the influence distance values, we find that the perturbation significantly influences the "Sports" subreddits.

3.2.4.3 Political Blogs Ranking with HITS

Graph-based ranking methods are widely used to rank webpages. Here, we consider a scenario where removing certain pages from a website (either intentionally by the website owner, or maliciously through shadow bans by an external party) can significantly change the rankings of other pages. Consider a political web forum where members post views and opinions on certain topics and issues. The search result rankings are determined by a graph ranking algorithm, and higher-ranked opinions are more likely to be read and shared. Here, one could imagine a nation-state actor that would want to promote biased content. The nation-state actor can create webpages to add various links in the graph structure, or even identify websites to shadow ban, which could manipulate the ranking results so that certain political opinions are more exposed to the public. By using the articles in the forum as the nodes and the hyperlinks between different articles as edges, the graph ranking methods can recommend popular articles in the forum based on the graph structure. However, there could be some nodes that are vulnerable and have high sensitivity indices with respect to the graph ranking method. They become the target for the attackers who wish to manipulate the article rankings and promote their own content. As such, blog managers and social scientists may wish to collaborate with ranking algorithm developers to make sure such ranking results are fair and stable with respect to potential perturbations (deletion of blogs).

In this case study, we explore the sensitivity of the HITS algorithm on the political blog dataset (Kleinberg 1999). The dataset includes a topic citation graph between liberal and conservative blogs prior to the 2004 U.S. Presidential Election. We subsampled 397 nodes (i.e, blogs) and 12,365 edges (i.e., hyperlinks between blogs), removing all nodes with degree less than 30. Our goal was to explore the structural changes in the network that result in drastic ranking changes in the HITS algorithm.

Identifying the Instance-level Sensitivity (T1, T3): Before the analysis, we made three assumptions about ranking manipulation: 1) the top-k items in the ranking are much more important than other nodes since readers typically only view the top results provided by the search engine. 2) It is riskier to manipulate a node with a higher rank since the readers may notice the changes. 3) To avoid having manipulations discovered, the attacker would assume a posture of minimum risk. As such, our goal is to discover how we can manipulate the ranking results by removing a node, while working under these constraints.

Based on our constraints, we apply selection rules to filter the sensitivity index table. We first click the ranking column to sort the ranking order. Then, we add the top-5 nodes to the protected nodes with constraints of *protect selected nodes from their ranking decrease by 0%*, which excludes all perturbations that would cause the rankings of these selected nodes to decrease. The constrained sensitivity index table is shown on the left, which contains 1) the ranking positions, 2) node names, 3) overall sensitivities, and 4) sensitivity details including positive/negative influence to liberal/conservative blogs. After sorting the rows by the sensitivity index column in decreasing order, a liberal blog, "liberaloasis.com" appears in the second row of the table. By observing the other columns, we find that “liberaloasis.com” is not in the top-10 rank; however, its removal can increase the rankings of the conservative blogs while decreasing the rankings of the liberal blogs.

Diagnosing the Perturbation Effects (T2): Next, we explore why this blog is so influential. After selecting “Explore the Perturbation in Detail” by clicking the cross button in the first column, all the details are listed on the right side of the interface (Figure 9). In the influence view, the radar chart indicates that there are 368 out of 397 nodes influenced by the removal of “liberaloasis.com”. 232 of the 368 nodes have their ranks increased while 136 decreased. The largest increase in the ranking positions is 16, and the largest decrease is 30. From the ranking change distribution (Figure 9 (6)), we find that most of the ranking changes happen in the range between 50 and 150. Since only mid-tier ranks are impacted, the effects of removing this node are subtle, meaning this change is not easily observable. However, the impact is significant. In the distribution changes view, we can observe that the median of the liberal blog ranking distribution decreases, while the conservative blog ranking distribution increases. By further exploring the proportions of both liberal and conservative blogs in the top-100 results, Figure 9 (5), we can see that the proportion of liberal blogs in the top-100 results has decreased by 5% (from 82% to 77%), thus subtlety shifting the site’s content.

In the ranking change distribution view, we also find that there are three liberal blogs with considerable ranking decreases after the perturbation. We further check the detailed view for information on the influenced nodes. We sort the original column to locate the exact ranking position and notice that the first three liberal blogs, “sununes”, “boloboffin”, and “elemming2”, have a large ranking decrease after the perturbation, which corresponds to the three liberal blogs in the ranking change distribution view mentioned above. We want to further explore the relationship between the ranking changes with the topological structure. The influence graph view (Figure 9 (8)) shows that the removed node influences a majority of liberal nodes, and as the influence distance increases, conservative nodes are indirectly influenced.

3.2.5 Evaluation and Expert Review

Along with our case studies, we have conducted a group interview with our collaborator (E0) and three additional domain experts in graph mining (E1, E2, and E3) to provide feedback on our framework. We began the interview by introducing our visual analytics framework, and the functionalities supported by each module. Then, we presented a demo of the analytical flow across the three previously described case studies. After this, the experts are allowed to freely explore the perturbation results of the three datasets (Facebook, Reddit, and Polblogs) over two ranking methods (Pagerank and HITS) in our system. The interview lasted approximately 90 minutes, and we collected free-form responses to the following questions:

1. Does the system meet the design requirements and address the analytical tasks proposed in our work?
2. Does our analytical pipeline match your daily workflow?
3. How is the information delivered through our system?
4. How would you perform the same tasks in conventional graph mining methods?

Framework: We received positive feedback from the experts in terms of our proposed framework. The experts found the framework to be practical with respect to the proposed problems. E1 appreciated that the framework is capable of handling the sensitivity issues that are related to nodes' attributes, and noted that such a framework can support not only graph-ranking developers but also experts in other fields evaluating whether the ranking algorithm is suitable for real-world ranking tasks. E2 appreciated our constraint filtering functionality since such an iterative analysis is one of his preferred approaches.

Visualization: The overall visualization techniques were also well received. E1 mentioned: *"With the knowledge of understanding what the system is capable of doing, I found most of the visualizations are straightforward and easy to understand. The newly designed influence graph is also intuitive once I learned what each encoding means."* E2 also appreciated that the interactive effect is helpful for understanding the ranking change effects after the perturbation. E3 suggested that we could add question-mark icons that link to descriptions for each view.

Limitations: The experts also identified several limitations of our current framework. E1 and E2 noticed that there is only one perturbation method supported in our system, which is the node removal. However, they all understood that the perturbation spaces (edge removal, node/edge addition) are far larger than the node removal space. E3 also mentioned that the visualizations may become crowded when the size of the influence graph is large.

Chapter 4

MACHINE LEARNING FAIRNESS ANALYSIS

In previous chapter, we find that the manipulation of data can cause not only the machine learning security issues, but also potential societal problems in real-world applications. In this chapter, a new framework has been introduced that focuses on fairness analysis in graph mining models. This framework addresses the growing concern about algorithmic bias and discrimination in machine learning models that rely on graph data. The fairness analysis framework provides a systematic approach to evaluate the fairness of graph mining models by examining the impact of sensitive attributes on the model's output. By integrating this framework into the development process of graph mining models, we can ensure that the models are fair and unbiased, which is crucial for making ethical and equitable decisions.

4.1 Fairness Analysis in Graph Mining Models

Common fairness definitions include *individual fairness* (Dwork et al. 2012) and *group fairness* (Pedreschi, Ruggieri, and Turini 2009; Pedreshi, Ruggieri, and Turini 2008), where individual fairness focuses on whether similar individuals are treated consistently and group fairness focuses on whether or not members of a protected class have the same probability of being assigned a positive outcome (for example, the same probability of receiving a housing loan). Difficulties arise due to the fact that *the sensitive attributes*[5] *vary from task to task.* Sensitive attributes may have commonalities across tasks, and there may be legally protected

[5]Sensitive attributes are generally defined to be traits of an individual which should not correlate with the algorithmic outcome, for example, gender, ethnicity, age, etc.

classes that need to be considered when measuring fairness. However, there is no single universal definition of fairness, and different applications of an algorithm may need to alter the definition of fairness depending upon the task at hand.

Recent work has focused on the development of a visual analytics framework designed to enable the exploration of multi-class bias in graph mining algorithms. The framework aims to answer critical questions in fairness. For example, how are biases reflected in the data as well as the outcome? Do debiased models accidentally impose other issues of fairness? The proposed framework is model agnostic, supports both group and individual fairness levels of comparison, and consists of a suite of interactive visualizations for investigating node attributes and topological features of graph elements to identify issues in algorithmic fairness.

From our literature review on fairness in graph ranking and visualization, we have identified several research challenges and gaps in the literature. These challenges were then evaluated with three data mining researchers who specialize in debiasing algorithms for graph learning models (two of which serve as co-authors on this paper). After iterative discussions with the experts, two major research challenges for auditing fairness in graph mining algorithms were identified:

Task-oriented Definitions of Groups. In conventional debiasing approaches, the definitions of protected groups may vary across applications. Typical examples include personal attributes associated with discrimination, such as gender, ethnicity, age, etc. However, identifying sensitive attributes and characterizing protected groups is a non-trivial task and demands expert knowledge to identify potential discrimination (Q. Wang et al. 2021). As such, there is a need for methods that can interactively define fairness, incorporate this definition into a debiasing method, and audit the impacts of the debiasing. In this paper, we use the term *group* to denote the protected groups characterized by sensitive attributes.

Trade-offs Between Group and Individual Fairness. Ideally, fairness adjustments to a machine learning model will maintain fairness between groups of nodes with similar attribute values. However, conflicting concepts of group and individual fairness (Binns 2020) can lead to cases where an algorithm that has been debiased at the group level now introduces bias at the individual level. Consider an employment recommendation system that meets the criteria for group fairness. Applicants in protected groups may receive more competitive rankings in order to keep statistical parity on selected attributes (such as gender or ethnicity). However, other candidates with similar abilities may now be de-ranked in order to ensure group fairness. Thus, it is crucial that algorithm designers have the means to explore individual and group fairness.

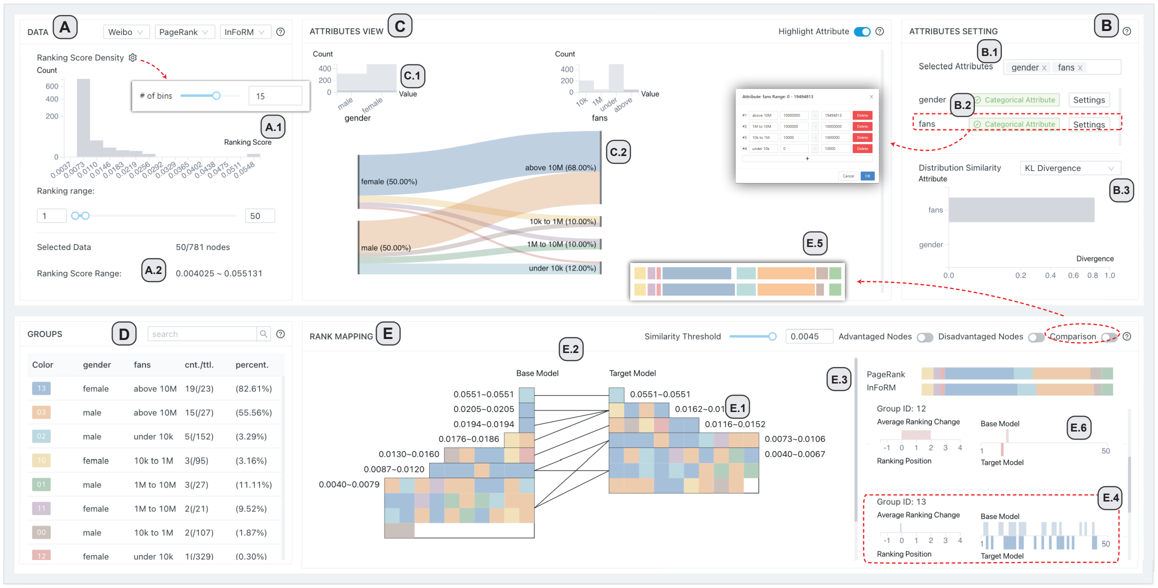
4.1.1 Analytical Tasks

We have also identified common ranking analysis and fairness auditing tasks that could benefit from a visual analytics approach. These tasks were refined through discussions with our co-authors who are the lead developers of several recent fairness aware graph mining algorithms.

4.1.1.1 T1: Define Target Nodes and Groups

Analysts should be able to specify sensitive attributes and inspect protected groups by defining:

- *T1.1:* Which portion of nodes are the most important, and;
- *T1.2:* Which attributes are critically important for fairness.



Selected Data 50/781 nodes

Ranking Score Range: 0.004025 ~ 0.055131

Selected Attributes: gender, fans

gender: Categorical Attribute — Settings

fans: Categorical Attribute — Settings

Group ID: 12

Group ID: 13

Color	gender	fans	cnt./ttl.	percent.
13	female	above 10M	19(/23)	(82.61%)
03	male	above 10M	15(/27)	(55.56%)
02	male	under 10k	5(/152)	(3.29%)
10	female	10k to 1M	3(/95)	(3.16%)
01	male	1M to 10M	3(/27)	(11.11%)
11	female	1M to 10M	2(/21)	(9.52%)
00	male	10k to 1M	2(/107)	(1.87%)
12	female	under 10k	1(/329)	(0.30%)

Figure 15. Fairness diagnosis of InFoRM (a debiased ranking model) on Weibo social network data. (A) The analyst selects the top-50 ranked nodes. (B) The analyst defines the "gender" and "fans" attributes as protected classes of interest in the attributes setting. (C) The attributes view shows that in the top-50 ranked nodes, the "gender" attribute ("female/male") is equally distributed. The view also shows the distribution of the "gender" attribute across the entire dataset (C.1), where it can be observed that "females" make up a larger portion of the entire dataset. The parallel sets portion of the attributes view (C.2) shows that nodes with "more than 10 million" followers make up the largest component of the top-50 nodes. (D) Selected nodes are grouped by the "gender" and "fans" attributes. (E) The rank mapping view shows group proportions (E.1) and supports comparison between raking algorithms by mapping the change (E.2) in each node's rank between the two ranking algorithms being explored. The group proportion view (E.3) shows few proportional changes when comparing the original ranking algorithm to the InFoRM model. The group shift view (E.4) shows that the average ranking of the "group 02" with attributes of "male" and followers "under 10 thousand" has increased by 2 positions, which may indicate that the InFoRM model has indirectly created a group preference.

4.1.1.2 T2: Reveal the Impact of Topological Structures and Attributes on Ranking Fairness.

Analysts should be able to diagnose the algorithmic fairness of graph ranking models by understanding the impacts of their topological structures and attributes on ranking fairness. Since the ranking results have no ground truth and are sensitive to changes in the graph structure (T. Xie, Y. Ma, H. Tong, et al. 2021), a base model is needed as a reference in order to explain the debiasing impact of a target model. Additionally, group fairness and individual fairness may have conflicting rule sets. When comparing models, analysts want to explore:

- *T2.1:* Which nodes are advantaged/disadvantaged by the model?
- *T2.2:* Which groups are advantaged/disadvantaged by the model?

4.1.1.3 T3: Diagnose Content Bias in Ranking Results

Display space is a bottleneck for showing all individual rankings. For example, Google searches list approximately 20 records per page, and the higher the rank, the more clicks. However, records listed on later pages may have similar relevance to the top ranked pages. This phenomenon has been studied by Pitoura et al. (Pitoura et al. 2018) which noted that *content bias* may occur when information is displayed in different ways. There two major analytical questions when diagnosing content bias:

- *T3.1:* Which nodes have similar relevance (ranking scores)?
- *T3.2:* What is each node's position in the ranking result, and how likely is it that content bias has occurred in similar nodes?

4.1.2 Design Requirements

Based on the analytical tasks, we engaged in an agile design process involving multiple iterations of the FairRankVis framework in collaboration with our domain experts. We have identified several design requirements and mapped different analytic tasks to each requirement.

4.1.2.1 D1: Visualize the Attribute Compositions of Target Nodes.

The system should support the selection of target nodes from the graph (T1.1). To enable the inspection of node attributes, the system should interactively visualize the composition of attribute values among selected nodes and visualize necessary metrics to assist analysts in selecting attributes for future diagnosis (T1.2).

4.1.2.2 D2: Visualize the Algorithmic Bias and Content Bias.

The system should visualize both algorithmic bias (T2) and content bias (T3) for selected nodes and attributes with the following views:

- *D2.1: Rank Mapping View*, which integrates ranking results that are mapped from the base model to the target model (T2.1) as well as the summary of nodes that have similar ranking scores (T3.1, T3.2).
- *D2.2: Group Proportion View*, which compares the proportional difference in terms of analyst-defined groups. The view should support a global proportion overview and a pair-wise proportion difference in terms of each group (T2.2).

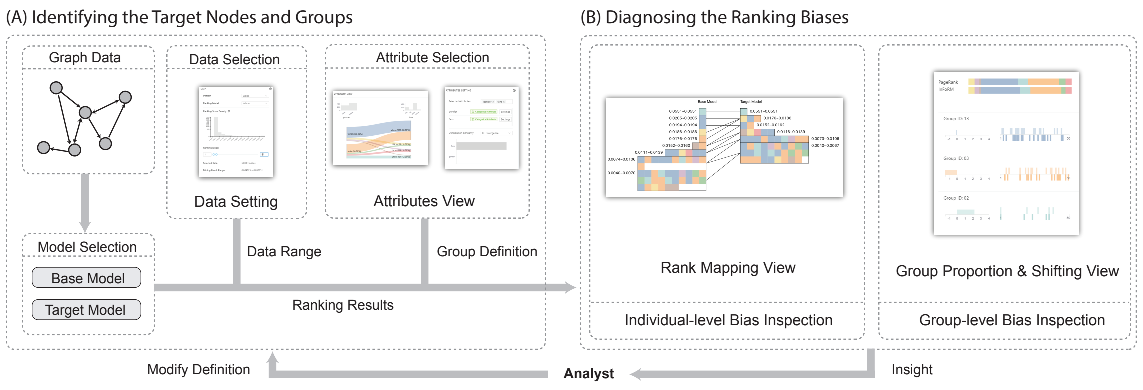


Figure 16. The FairRankVis Framework consists of two stages: (A) the identification of target nodes and groups stage, and (B) the diagnosis of biases in ranking results stage. In stage (A), the analyst can select the base model and the target model to be inspected. The ranking results will be generated after model selection. The analyst then defines a range of nodes, either the top-k nodes or nodes who have similar ranking scores, and then defines the groups based on selected attributes. (B) The analyst can then explore and inspect both individual-level and group-level bias. The framework also supports modifying the definition of fairness at any time during the analysis process.

- *D2.3: Group Shift View*, which shows how analyst-defined group rankings shift from the base model to the target model (T2.2).

4.1.3 Visual Analytics Framework

Based on the analytic tasks and design requirements, we have developed a visual analytics framework (Figure 16) to support fairness auditing in graph-based ranking algorithms. The framework is designed to first load the graph data and then compute the ranking results using the analyst selected targeted model and base ranking model (Figure 16 A). Then the analyst can interactively define the target attributes for fairness auditing (Figure 16 B). As the definition of group and target nodes are updated by the analyst, the ranking results are updated across all views to support bias inspection. Analysts can modify the group definitions at any time to explore issues of algorithmic fairness.

The framework supports two major functionalities: 1) identifying the target nodes and groups, and 2) diagnosing potential ranking biases. By identifying the target nodes and groups, the analyst can select a portion of nodes according to their specific analytical goals and explore the attribute distributions. The selected nodes are automatically categorized by the analyst-defined groups. The analyst can also explore the ranking results of both the base ranking model and the target ranking model to explore group/node shifts, proportions, and distributions of similar nodes. The analyst can flexibly modify the definition of a group at any time to explore both single and multi-attribute fairness. Our modular design enables analysts to freely integrate any graph-based ranking models for use as the target or base model. For demonstration purposes, we apply PageRank as the base model and AttriRank and a debiased PageRank (InFoRM) as the target models.

4.1.3.1 Background of Graph Ranking Models

AttriRank (Hsu et al. 2017) is a PageRank-based model that uses the topological information and node attributes to compute the ranking vector $\boldsymbol{r}$:

$$\boldsymbol{r} = c\boldsymbol{Q}\boldsymbol{r} + (1-c)\boldsymbol{P}\boldsymbol{t} \tag{4.1}$$

where

$$P_{ij} = \begin{cases} \frac{1}{\delta_j}, \text{if directed edge}(j,i) \in E \\ \frac{1}{N}, \text{if} \delta_j = 0 \\ 0, \text{otherwise} \end{cases} \quad , Q_{ij} = \frac{s_{ij}}{\sum_{k \in V} s_{kj}} \tag{4.2}$$

δ_j denotes the out-degree of node j, and s_{ij} the degree of similarity with respect to the attribute values of the nodes. In AttriRank, the Radial Basis Function (RBF) kernel is defined as the similarity measure:

$$s_{ij} = e^{-\gamma ||x_i - x_j||_2^2} \tag{4.3}$$

where γ denotes the distance influence. In this way, external attribute values are integrated into the ranking procedure which is more robust for handling nodes that have missing edge information.

InFoRM (J. Kang et al. 2020) is a generic individual fairness framework for quantitatively measuring the potential bias in graph mining tasks including graph ranking, clustering and graph embedding. The InFoRM framework can perform three types of debiasing methods including (1) debiasing the input graph, (2) debiasing the graph mining model, and (3) debiasing the mining result. We employ InFoRM to debias the ranking results of PageRank to simulate a situation where the debiased model does not have access to the input data and the model. Mathematically, this process is realized with the following objective function:

$$Y^* = argmin_Y J = ||Y - \bar{Y}||_F^2 + \alpha Tr(Y^T L_S Y) \tag{4.4}$$

where Y^* denotes the debiased ranking result, $\bar{Y}$ denotes the original ranking result. $\alpha > 0$ is the regularization parameter, and L_S is the Laplacian matrix of the similarity matrix S[6]. This equation minimizes the sum of the squared Frobenius distance between ranking results and the regularized tethnicity of the matrix produced by $Y^T L_S Y$ so that both the difference of the ranking results before and after debiasing (Y and $\bar{Y}$) and the bias (defined as $Tr(Y^T L_S Y)$) are minimized.

4.1.3.2 Identifying the Target Nodes

Our framework is designed to enable a flexible definition of ranks and attributes to be considered when diagnosing fairness. Recent research (chartier2011sensitivity; Zehlike_2017; Yang and Stoyanovich 2017) emphasizes that the top-k elements will receive more attention,

[6]The similarity matrix S uses cosine similarity and Jaccard similarity.

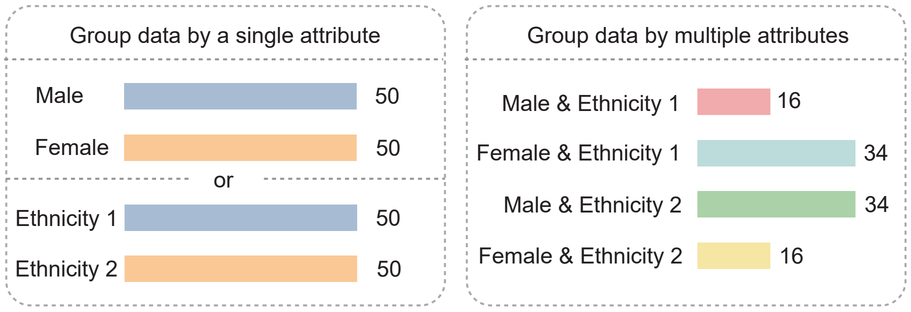


Figure 17. Different group definitions can lead to different fairness insights. Suppose there are 100 nodes who have similar ranks. If we group the nodes only by ethnicity or gender, the proportions are equal, which might imply that the outcome is fair in terms of both ethnicity and gender. However, if we group the nodes by both ethnicity and gender, we may find potential inequalities at the intersection of the two attributes.

and ranking bias is typically explored with respect to the top-k ranks. In our proposed framework, a data setting panel (Figure 15.A) is configured to enable the analyst to select the top-k nodes. This is facilitated by the *Ranking Score Density Histogram* (Figure 15.A.1), which shows the ranking score distribution for the target ranking model. The analyst can interactively modify the number of bins by clicking the gear icon, and the histogram supports brush selection to select a specific ranking range *(T1.1)*. For example, if the analyst cares about potential biases of nodes who have similar ranking scores, then the analyst can brush a particular bin on the histogram and all the nodes within that ranking score range are selected. If the analyst wishes to select a specific ranking position, a slider is configured to enable the analyst to select nodes from rank m to n. In this way, the analysts can explore how attributes are distributed for any specific range of ranks.

4.1.3.3 Defining Groups

Once a range of nodes is selected, the analyst is able to explore attribute information and define groups through the attribute setting panel (Figure 15.B) and attribute view (Figure 15.C). Recent work (Dwork et al. 2012; Kilbertus et al. 2017) suggests that a general fairness principle is based on whether *similar nodes will have a similar ranking*. In other words, defining a group means defining individuals that are similar. Wang et al. (Q. Wang et al. 2021) note that the definition of similarity is not easy to obtain and may vary from task to task. The wrong definition of similar nodes can lead to wrong conclusions with respect to bias and fairness. Figure 17 shows a simple example of this phenomenon: nodes who have similar ranks are distributed evenly if we only group them by either ethnicity *or* gender. However, the data reflects a disproportionate distribution when we group the nodes by ethnicity *and* gender. Our framework enables analysts to explore all available attributes and across combinations of attributes. In our framework, the analyst selects one or more categorical attributes, and each combination of category is now considered a group. From the example in Figure 17, if the analyst selects gender and ethnicity, there would be four groups to be audited for fairness.

Attributes View. To support the interactive definition of groups *(T1.2)*, we have designed an attribute setting panel (Figure 15.B) and an attribute view panel (Figure 15.C). The attributes view panel employs a parallel set where each selected attribute is visualized with multiple bars. Selected nodes are encoded as curves with different widths. Both the height of bars and the width of the curves encode the number of nodes mapping to a specific attribute value. Additionally, the distribution of attributes across the selected nodes is visualized with a histogram (Figure 15.C.1). We use a light grey color to show the attribute distribution for the entire dataset, and the dark grey color histogram shows the distribution of attributes for the selected nodes. The attribute setting panel (Figure 15.B) enables the flexible selection of one or more attributes by clicking on the multiple selection area (Figure 15.B.1). All corresponding views including the attributes view (Figure 15.C), the group table view (Figure 15.D), the rank mapping view (Figure 15.E), the group proportion view (Figure 15.E.3) and the group shift view (Figure 15.E.4) are automatically updated as the selected attributes are changed. Since group fairness is most often based on categorical attribute values, we also include a customization feature that allows analysts to categorize attributes that may have continuous values. For example, protected classes for age are often grouped into ranges, e.g. under 18, 65+, etc..

We also provide another histogram (Figure 15.B.3) to facilitate the comparison of distribution similarities on selected attributes between selected nodes and the entire dataset. The metric for measuring distribution similarities can be customized based on the analysts' needs. Currently, the framework supports Kullback-Leibler divergence for demonstration purpose. The height of the bars are mapped to the differences of the between the distributions of the selected nodes and the entire dataset on a specific attribute.

4.1.3.4 Diagnosing the Ranking Biases

Once the nodes are selected and sensitive attributes defined, the corresponding groups are automatically generated, assigned a label and unique color, and displayed in the group table (Figure 15.D). Once groups are defined, the fairness audit can begin. Here, it is important to note that biases in machine learning models can arise due to issues with the *Data* and/or issues with the *Model*.

Diagnosing Data Bias. Real-world data can be either insufficiently sampled or reflect existing prejudices. Thus, it is inappropriate to ask models to be fair when being optimized on biased data. In terms of graph ranking, it is critical to understand how groups are distributed prior to applying a debiased ranking model. Our system first ranks the data with what we refer to as the *base* model. For demonstration purposes, we employ PageRank as the base model.Exploring the base model can help reveal the underlying topological features of the data. What we are interested in is if there are already signs showing disproportional distributions for each group. From the base model ranking, we can explore whether certain groups have higher ranking scores than others. For example, if the base model (PageRank) shows that when evaluating node ranking based on gender, nodes that are marked as male are ranked relatively higher than female nodes, then other PageRank-based models are very likely to observe a similar distribution between the male group and the female group. In this case, the gender bias is not inherited from the model but the data.

Diagnosing Model Bias. Our framework supports diagnosing three types of bias: Content *(T3.2)*, Group *(T2.2)* and Individual Bias *(T2.1)*.

1. *Content Bias.* In real-world applications, a full ranking of millions of items simply cannot be displayed, and is typically culled to some top-k rank. In this setting, even the nodes who have the same ranking scores can have a large difference in ranking positions, and this problem is referred to as content bias. For example, imagine a list of items where the second through the seventh item have identical ranking scores. The method of display implies inequality in ranking even though ranks two through seven have equal ranking scores. Here, the implicit ordering can lead to significant differences in their exposure rates. To help analysts explore this phenomenon, we group nodes into clusters based on their ranking scores *(T3.1)*. Algorithm 4 shows the k-means-based clustering algorithm. The idea of the clustering algorithm is to group as many nodes as possible into a cluster such that the maximum difference between ranking scores in this cluster is less than the analyst-defined similarity threshold. The algorithm outputs the minimum number of clusters to satisfy the analyst-defined rules of

similar ranking scores. The analyst can inspect the cluster for signs of possible content bias.

2. *Group Bias.* Many fairness metrics have been proposed for measuring group fairness (Hardt, Price, and Srebro 2016; Dwork et al. 2012). These methods attempt to measure the degree of discrimination or bias (Tutorial99:online). However, there is no single term that universally represents bias. We denote *group bias* as the bias that reflects the ability of the model to achieve statistical parity between groups, where a group is defined with respect to the analysts' selected sensitive attributes. The goal of the framework is to enable analysts to audit whether the ranking results of a model exhibit direct or indirect preferences towards one or more groups, resulting in lower ranking scores for the disadvantaged groups. Compared with the content bias, where disadvantages can be due to display constraints, group bias can be mitigated algorithmically. To observe the impact on groups' ranking between the base and the target model, we formalize the ranking changes for each group by computing the average ranking position change *(T2.2)*:

$$\Delta_g = \frac{1}{n}\Sigma_{v \in V_s, v \in g}(r'_v - r_v), \tag{4.5}$$

   where Δ_g is the average ranking change of group g among selected nodes V_s, r'_v is the ranking position of node v in the target model, r_v is the ranking position in the base model, and n is the number of nodes in both the selected nodes and group g.

3. *Individual Bias.* Individual bias represents how the model guarantees that nodes with similar attributes will receive similar rankings. It is important to understand if individual nodes have been "sacrificed" or privileged by the model in order to reduce group bias. To help analysts explore the individual biases among selected nodes, we label the

selected nodes as advantaged/disadvantage nodes according to their ranking position changes (increase/decrease) between the base and target model *(T2.1)*.

Algorithm 4 Clustering Similar Ranking Scores

```
1: Inputs: similarity threshold $\delta$; selected nodes $V$;
2: Outputs: clusters $C$ with maximum ranking score difference $d \leq \delta$
2: for $k$ in range (1, $V$.length) do
3: $\mathbf{C} \leftarrow k_{means}(k, V)$
3:     if $d_c <= \delta, \forall c \in \mathbf{C}$ then
4: return $\mathbf{C}$
4:     end if
4: end for
5: Return $\mathbf{C}$ =0
```

Rank Mapping View. The rank mapping view (Figure 18) consists of two columns of stacked rectangles, where the left column shows the ranking results of the base model, and the right column shows the ranking results of the target model. For each column, small squares that represent nodes of the analyst-defined groups are organized into large rectangles, where each rectangle represents a cluster (from Algorithm 1) that contains nodes with similar ranking scores *(T3.1, T3.2)*. From top to bottom, the nodes are ranked from m to n, and in a cluster, the nodes are mapped from left to right according to their rank (high to low). Each cluster from the base model is connected to a corresponding cluster from the target model by a grey line when they share the same node(s), which illustrates how the ranking of this node changes between models *(T2.1)*. The color of the square maps to the analyst-defined groups. In this example, we can observe that members in group 1 have a relatively higher rank position than those in group 2 from the top-1 to top-10 ranks, In Figure 18, we can observe that there are four nodes belonging to a cluster with a ranking score from the base model ranging from 0.123 to 0.124. Three of the nodes are in "group 1", while only one is in "group 2". The node from "group 2" has been ranked in the sixth position. This means that even though their ranks are functionally equivalent, the node belonging to "group 2" will likely have a lower exposure rate than equivalent nodes in "group 1", indicating that content bias may occur.

Such phenomena can be significant when the size of the cluster is larger. Imagine a cluster with 30 nodes whose ranking scores are functionally equivalent. The 30th node is so far below the 1st node of this cluster that the differences in exposure are extremely uneven. Such content bias is inevitable given the traditional ranked list displays in real-world applications; however, the analyst should at least be aware of any content bias and can consider modifications to the display list to adjust for such biases. The rank mapping view also supports rich interactions to facilitate analysts to discover more information. There are also switch buttons that allow the analyst to highlight advantaged/disadvantaged nodes (Figure 15.E.5), and analysts can hover on a single node to see the same node in the ranking result of another model. The tooltip is used to show node attributes on mouseover. If analysts click on a single node, the view will highlight all nodes in the corresponding cluster and their corresponding ranking positions in the debiased model.

Group Proportion View. The group proportion view is designed to illustrate the target ranking model's effects on each group's proportion *(T2.2)*. The group proportion view consists of two sets of bars and each set shows the composition of selected nodes sorted by both ranking models respectively (Figure 19). To facilitate inspection, we support switching the view mode between the proportion mode and the comparison mode. The proportion mode displays the stacked bars to summarize the overall group distribution of the selected nodes, while the comparison mode supports a direct comparison of group proportions between models. In other words, the comparison mode helps analysts perform pair-wise comparisons of the same group proportions between different models. Analysts can toggle between the

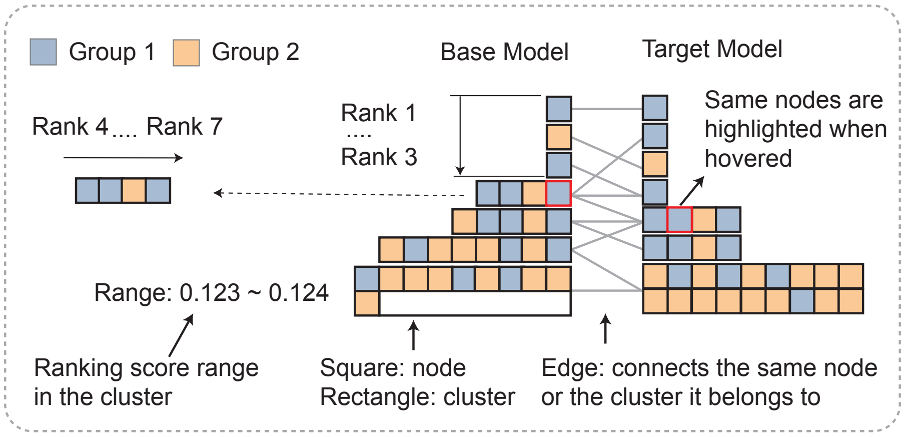

Group 1
Group 2
Base Model
Target Model
Rank 4 .... Rank 7
Rank 1
....
Rank 3
Same nodes are highlighted when hovered
Range: 0.123 ~ 0.124
Ranking score range in the cluster
Square: node
Rectangle: cluster
Edge: connects the same node or the cluster it belongs to

Figure 18. Visual encoding of the rank mapping view. The ranking results of the base model and target model are listed separately. Small squares represent nodes and are colored with respect to the analyst-defined groups. These squares are organized into large rectangles, and each rectangle represents a cluster that contains nodes with similar ranking scores. From top to bottom, the nodes are ranked from m to n (in the example $m = 1$ and $n = 30$), and, in a cluster, the nodes' ranks from high to low are mapped from left to right. Each cluster from the base model is connected to a corresponding cluster in the target model by a grey line when they share the same node(s).

proportion and comparison mode by using the switch button on the right side of the title bar of the rank mapping view (Figure 15.E).

Group Shift View. The group shift view (Figure 20) is designed to inspect both group bias *(T2.2)* and individual bias *(T2.1)*. For inspecting group bias, the bar chart on the left shows the average ranking change of each group. The color of the bar encodes the identity of the group. The bar chart on the right shows the distribution of group members in the base model and target model, and the analyst can diagnose group shifts in selected nodes to understand the corresponding fairness trade-offs between models. For inspecting individual bias, analysts can hover on the squares of the ranking mapping view to trigger the highlighting of that node on the right side of the group shift view. This can help analysts to understand how individual bias occurs when applying debiased algorithms to achieve group fairness.

Interactions. Our framework employs multiple coordinated views to allow analysts to inspect group, individual, and content biases. These views are supported by a set of rich interactions. The selection of data in the data summary view (Figure 15.A) directly updates the content of the attribute view (Figure 15.C) and the rank mapping view area which includes the group proportion view and the group shift view (Figure 15.E). The attribute view (Figure 15.C) is dynamically updated based on selected attributes in the attribute setting panel (Figure 15.B), and selections in the attribute setting panel also updates the colors of the entire system as the colors encode the analyst-defined groups. For the rank mapping view (Figure 15.E), analysts can adjust the similarity threshold slide bar to define how nodes are clustered based on the ranking scores, and analysts can toggle advantaged/disadvantage nodes to highlight nodes that have the rank increase/decrease. Analysts can also hover over squares in the rank mapping view to highlight and locate the node's position in both the vanilla and debiased algorithm, and tooltips are used to provide details of the node attributes. Along with hovering, analysts can also click on a node to show how the ranking positions of all nodes in the cluster change from the base model to the target model. Finally, analysts can toggle the comparison model to enable pairwise comparison of the same group between two models as shown in Figure 15.E.5.

4.1.4 Usage Scenarios

In this section, we present two usage scenarios to demonstrate how our framework supports fairness audits in graph-based ranking models. We first show how graph ranking model developers analyze the potential bias in AttriRank model. Next, we illustrate how fair rank-

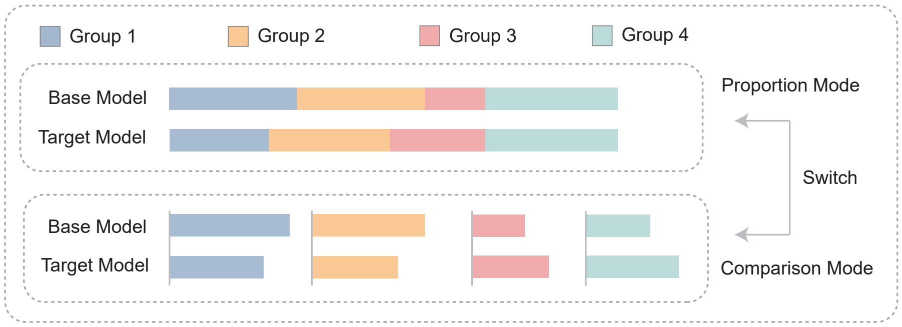


Figure 19. Visual encoding of the group proportion view. Two sets of bars are visualized to show the group proportion among selected nodes for the base model and target model. The group proportion view has two modes: proportion mode and comparison mode. Proportion mode uses a stacked bar to show the overall group distribution within a model. The comparison mode splits the bars by group for between models.

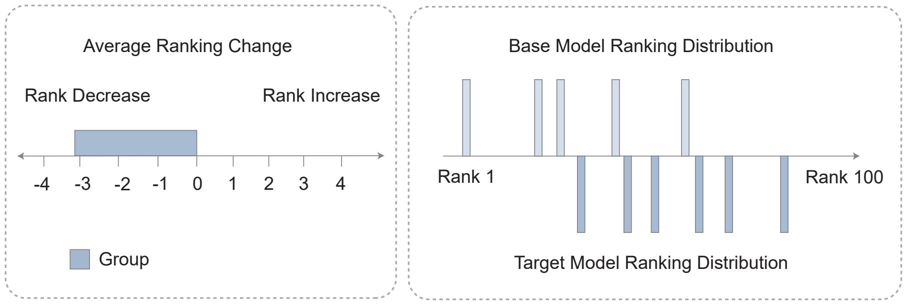
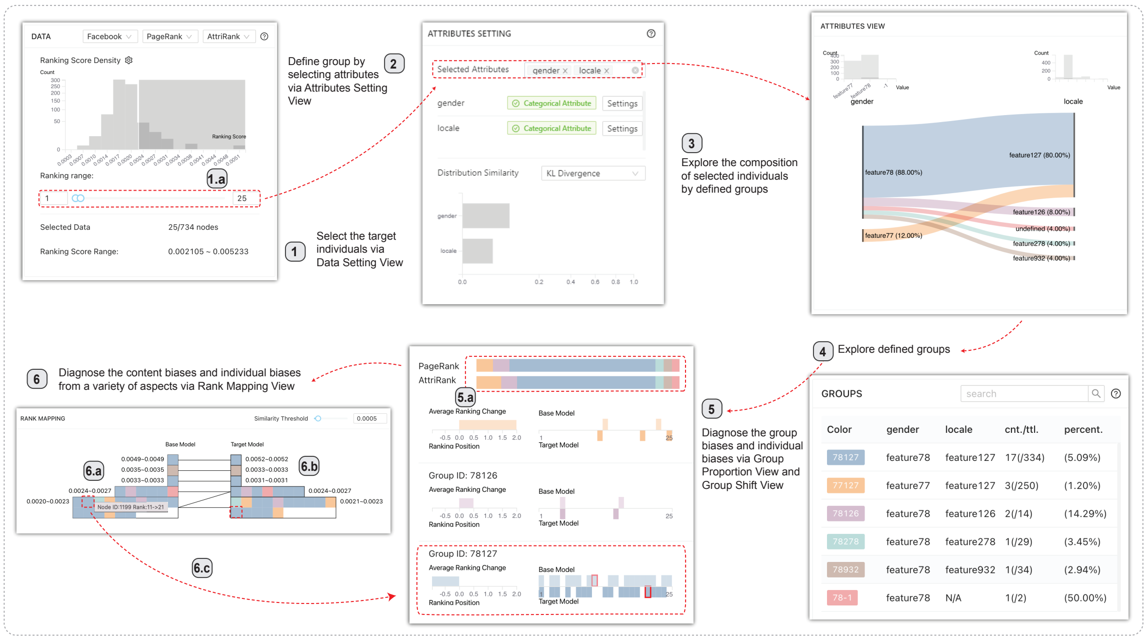


Figure 20. Visual encoding of the group shift view. The bar chart on the left shows the average ranking change of each group. The color of the bar encodes the identity of the group. The bar chart on the right shows the distribution of group members in the base model and target model, where the x-axis maps to the ranking position of selected nodes.

ing model developers inspect the trade-off by applying a debiased ranking model (InFoRM). Finally, we report on an expert review of the system conducted with four domain experts.

4.1.4.1 AttriRank Bias Inspection on Facebook

In social network analysis, ranking algorithms utilize an account's topological structure and demographic information for a variety of tasks including link predic-


1 Select the target individuals via Data Setting View

2 Define group by selecting attributes via Attributes Setting View

3 Explore the composition of selected individuals by defined groups

4 Explore defined groups

5 Diagnose the group biases and individual biases via Group Proportion View and Group Shift View

6 Diagnose the content biases and individual biases from a variety of aspects via Rank Mapping View

GROUPS

Color	gender	locale	cnt./ttl.	percent.
78127	feature78	feature127	17(/334)	(5.09%)
77127	feature77	feature127	3(/250)	(1.20%)
78126	feature78	feature126	2(/14)	(14.29%)
78278	feature78	feature278	1(/29)	(3.45%)
78932	feature78	feature932	1(/34)	(2.94%)
78-1	feature78	N/A	1(/2)	(50.00%)

Figure 21. AttriRank Bias Inspection on Facebook. (1) We select the top-25 nodes with ranking scores ranging from 0.002105 to 0.005230. (2) We avoid selecting attributes that are suppressed for most nodes, and choose to use "gender" and "locale" as our sensitive attributes to divide nodes into groups. (3, 4) We notice that the "group 78127" in which members have "gender" value "feature 78" and locale value "feature 127" has the largest proportion, and (5) the group proportion view shows that a large portion of "group 78127" is found in the top-k ranking results from both models. (6) From rank mapping view, we find that cluster (6.a) and (6.b) contains nodes with similar ranking scores from 0.0024 to 0.0027, while these nodes have different ranking positions the difference in ranking scores is less than the analyst-defined threshold $\epsilon = 0.0005$, which has implications for content bias. (6.c) Finally, we find that node "1199" experiences a large ranking drop from 11th to 21st, which has potential implications for individual fairness. For group bias, we find that "group 78127" in the group shift view (6.c) was negatively influenced by AttriRank, with the average ranking and top-k proportion decreasing when compared to their rankings from PageRank.

tion (Gleich 2015), advertising (Heidemann, Klier, and Probst 2010) and recommendation (Gori:2007:IRB:1625275.1625720). Biases in rankings can have a huge impact with regard to content exposure, personal opportunities, and business strategies. As such, auditing the ranking algorithms used in such systems can help analysts understand whether the ranking results can comprehensively be considered to be unbiased under a variety of fairness definitions. For example, in the recommendation-based social network application, if accounts of male users are more likely to be recommended than female users, those male users will have more opportunities for content exposure and networking opportunities. Even in the case where male and female users have equal rankings, their level of exposure might still be affected by the ranking position arrangement. Here, content bias can effectively drive more clicks to the top-1 account, when, in reality, the top-10 account may have an equal ranking. Furthermore, when exploring group-level fairness, single attribute fairness audits may show that results are balanced. However, the intersection of sensitive attributes, e.g. gender, ethnicity, age, might reveal further biases in the rankings as it is possible for a ranking model to learn biased patterns both implicitly or explicitly so that certain groups are treated with advantages while others are disadvantaged. In this usage scenario, we audit AttriRank (Hsu et al. 2017) when applied to a Facebook social network dataset (NIPS2012_7a614fd0). The data is subsampled to a subgraph with 734 nodes and 74254 edges. Each node has 24 attributes that describe the demographics of a user. All identifiable information is anonymized and some attribute values are suppressed. In this usage scenario, we assume that the model developers have no prior knowledge about the data.

Identifying the Target Nodes and Groups (T1): The data setting view displays the ranking score density distribution (Figure 21.1.a). The majority of the nodes have a ranking score ranging from 0.0017 to 0.0020. The analyst selects the top-25 nodes to explore the results of

AttriRank. The analyst inputs 25 into the right-hand input box of the ranking range section, and the bottom of the data setting view shows the information of the selected nodes. Next, the analyst explores the attribute distributions in the attributes view and see that attributes political and region have been suppressed for the majority the nodes. The analyst chooses to remove such attributes from the analysis. Among the top-25 nodes, the analyst finds that there are two attributes with heavily non-uniform distributions: (1) the ratio of the *gender* value, which has two classes - "feature 78" and "feature 77", and the ratio between the two classes is 88% to 12% respectively. (2) The *locale* has five classes, and the selected nodes with the *locale* value of "feature 127" make up a greater portion of the dataset than other *locale* values (Figure 21.3). The analyst then select *gender* and *locale* to serve as the sensitive attributes that form the basis of our fairness audit.

Diagnosing the Ranking Biases (T2): After selecting *gender* and *locale* as the criteria for defining target groups, all possible groups are generated and displayed in the groups view as shown in Figure 21.4. Among the 6 generated groups, the analyst identified that "group 78127" (*gender* value "feature 78" and *locale* value "feature 127") has 16 members in the top-25 ranks, thereby occupying the majority of the top-k ranks. Given the disproportionate representation by "group 78127", the analyst decides to further explore the effects that AttriRank had on the ranking distributions compared with the base model (PageRank). By exploring the group proportion view (Figure 21.5.a), the analyst observes that "group 78127" was also disproportionately favored in the top-25 rankings by the base model, PageRank. This indicates that *the reason that the nodes in* "group 78127" have a higher rank is due to their topological features as opposed to the attribute rank based adjustments from AttriRank. The group proportion view also shows that the proportion for each group in the top-25 rankings saw no significant changes between the PageRank and AttriRank rankings, with only "group 77127" and "group 78127" seeing small changes in representation.

Next, the analyst inspects for content bias in the ranking results (Figure 21.6). Here, the analyst considers nodes with ranking scores that are within $\epsilon = 0.0005$ of each other to have the same rank and sets this threshold number as the similarity threshold. By inspecting the rank mapping view, the analyst observes that the top-25 nodes can be grouped into 5 clusters for both PageRank and AttriRank. The top-3 nodes have substantially different rankings and form 3 unique clusters in both ranking models. For the remaining clusters, two clusters (the fourth ones) of both models in Figure 21.6.a and Figure 21.6.b cover the same ranking score range from 0.0024 to 0.0027. In other words, nodes in these clusters have approximately the same relevance or utility. However, their ranking positions range from 4th to 9th in PageRank and 4th to 10th in AttriRank, indicating that content bias is occurring and it is slightly more pronounced in AttriRank than PageRank.

Finally, the analyst inspects the effect of AttriRank's behavior on the top-25 nodes. By exploring nodes of rank mapping view (Figure 21.6.c), the analyst finds that node "1199" experiences a significant ranking drops from 11th to 21, which indicates that the AttriRank sacrifices the node during the ranking process, which may lead to individual bias. From the group shift view (Figure 21.5) the analyst also observes that the "group 78127" is the only group that has an average ranking decrease. AttriRank is designed to adjust rankings such that nodes with similar attributes have similar ranking scores; however, this optimization may bias the results towards specific groups. Thus, auditing tools, such as FairRankVis, can help analysts evaluate tradeoffs between algorithms, inspect for biases, and audit fairness definitions.

4.1.4.2 InFoRM Bias Inspection on Sina Weibo

In the second usage scenario, the analyst compares a debiased ranking model (InFoRM (J. Kang et al. 2020)) to the vanilla version (PageRank (Page et al. 1999)) and explores tradeoffs between individual and group fairness. As described in Section 5.1, overemphasizing group fairness can propagate issues of individual fairness, and it is difficult to balance the ranking positions to guarantee both group fairness and individual fairness. As such, it is necessary for ranking model developers and fairness researchers to understand the trade-offs of a debiased ranking model when applied to a given dataset. Here, the analyst explore a social network dataset collected from Weibo (UCIMachi16:online) where each node consists of four social attributes (*gender*, *fans*, *account level*, and *location*) about the demographic information of a Weibo user. For demonstration purposes, we subsampled the data down to 781 nodes and 2315 edges. Again, the analyst has no prior knowledge about the dataset.

Identifying the Target Nodes and Groups (T1): After the models and the dataset are loaded, the analyst inspects the Ranking Score Density Histogram and observes that the nodes are concentrated at a ranking score of around 0.0073 (Figure 15.A.1). The analyst is interested in how top-k nodes with different ranking scores are affected by InFoRM. The analyst selects the top-50 nodes as the target nodes. In the data setting view (Figure 15.A.2), it can be observed that most of the ranking scores for the selected nodes lie in the range between 0.004025 and 0.055131.

Then, the analyst explores the attributes in the attributes view. Here, the analyst observes that the proportions of "males" and "females" are nearly identical, i.e., 50%:50% (Figure 15.C.2). However, the global *gender* distribution on the entire dataset shows a completely different pattern where the proportion of "females" is larger than "males" (Figure 15.C.1). Next, when inspecting the attribute *fans*, which describes the number of followers for the user, the attributes view shows that the majority of users (88%) have over 10 thousand followers, resulting in mismatched distributions between the selected group and the entire dataset (Figure 15.C.2). Since these two attributes show contrasting proportions between the full dataset and the top-50 nodes, the analyst decides to explore the ranking effects of nodes who have the same *gender* class and the same *fans* class. The group table view shows that there are 8 groups generated by this split, and the analyst finds that the nodes with "more than 10 million" followers have the largest population in the top-50 rankings (Figure 15.D). Furthermore, most of the "female" users (82.61%) and the "male" users (55.56%) who have "more than 10 million" followers appear in the top-50 user list.

Diagnosing the Ranking Biases (T2): To further understand how groups are affected by a debiased ranking model which focuses on maintaining individual fairness, the analyst first inspects how groups are distributed among the top-k nodes. By exploring the group shift view (Figure 15.E.6), the analyst observes that the "group 13" (representing "female" users who have "more than 10 million" followers) tends to have higher rankings than "group 03" (representing "male" users who have "more than 10 million" followers). The analyst wonders whether it is the target model that favors "group 13" by increasing the ranking scores of the nodes in "group 13". By observing rank mapping view (Figure 15.E.2), the analyst finds the "group 13" also has higher rankings than the "group 03" when nodes are ranked by the PageRank model. This indicates that "group 13" receives better rankings in both models and is not favored only by the target model.

The analyst further inspects the changes of the group's proportions in the group proportion view (Figure 15.E.3), and the analyst observes the proportion of groups are quite similar between ranking results in PageRank and those in InFoRM. By toggling the comparison mode to enable pair-wise comparison, the analyst finds that the proportion of "group 13" has slightly increased, and the proportion of "group 00" (representing male users who have followers between 10 thousand and 1 million) slightly decreased. Other groups distribution across the top-50 rankings maintain relatively the same proportion. When observing the group shift view, the analyst finds that "group 03"'s average ranking decreased by 1 position and "group 02" (representing "male" users who have "less than 10 thousand" followers) has its average ranking increased by 2 positions.

Here, the analyst wants to inspect the content bias of the ranking results from the target model. By tweaking the similarity threshold, the analyst finds that given the similarity threshold 0.0035, the top-50 nodes are clustered into 6 clusters (Figure 15.E.1) and each cluster has relatively more nodes compared with clusters of the base model. This indicates that the debiased ranking results tend to manipulate nodes to have similar ranking scores and reduce the potential for individual bias. However, this results in a larger content bias.

4.1.5 Evaluation and Expert Review

To further evaluate our framework, we conducted an interview with our collaborators (E0, E1), two domain experts (E2, E3) in graph mining and two domain experts (E4, E5) in machine learning and artificial intelligence. For the interview, we first introduced our system by describing the analytical tasks supported in the framework. We then demonstrate the components of our framework by walking through one of the two usage scenarios described in Section 4.1.4.1 and 4.1.4.2. Finally, the experts were allowed to freely explore the two datasets in the usage scenarios. The duration of the interview was approximately 90 to 100 minutes. On average, experts spent approximately 7 to 10 minutes to master the system and were able to explore bias information based on their own. All experts were able to fully understand the major functionalities of the system by asking a few questions during the exploration phase. After the free exploration stage was finished, we collected feedback from the experts using the following questions:

1. How well are the proposed analytical tasks supported with our design? (*Q1.*)

2. What are the traditional ways of addressing such tasks in conventional fairness audits of graph mining models? (*Q2.*)
3. How effective is this framework in supporting fairness audits? (*Q3.*)
4. How would the framework fit into your development circle? (*Q4.*)
5. Please rate the user experience from 1 to 5 (poor to good) considering the intuitiveness of the views, interactions and effectiveness of the analytic workflow. (*Q5.*)

Framework: We received positive feedback on the overall design of the framework. The experts noted that it is necessary to have such a framework to explore and identify fairness issues in graph mining models. E0 and E1 considered that the flexibility in defining target nodes and groups vastly facilitates the task-oriented analysis by swapping the nodes and groups on the fly. E1 appreciated the design of the rank mapping view, especially the support for individual-level bias inspection. *"Usually, only an aggregated measure is reported for the individual biases on all the nodes in a graph, and we also have to visualize the biased result for each node to fully obtain the information of individual bias (Q2).* With the help of interactive visualizations, we can clearly observe the biases for each node in a detailed manner, which benefits the in-depth analysis and reasoning of fairness issues in different ranking algorithms. (Q3)" E2 and E5 appreciated that the framework fits the general process of fairness auditing since the fairness issues have attracted much attention; however, the definition of fairness is always controversial. Thus, by enabling an interactive definition of sensitive attributes, this framework can support a quick reanalysis of fairness under different constraints. (Q4)

Visualizations: The experts all agreed on the effectiveness of the visualization design in our framework (Q1). They noted that the combination of rank mapping view, group proportion view and group shift view can illustrate the impact of the ranking models on defined groups and nodes. E2 noted that the two modes of the group proportion view can reveal information in both group proportions and group-wise comparisons between models. E3 appreciated the design of the rank mapping view which depicts both individual bias and group bias simultaneously. *"This view could assist us in checking how effective the debiasing methods can be. The result can be easily observed in the rank mapping view." The average score for the user experience question is 4 out of 5 (with the lowest score being a three and the highest a 5) (Q5). Experts agreed that the workflow is clear and was enthusiastic about the ability to flexibly define protected groups. All of our domain experts felt that the interactions were appropriate and provide the necessary information.*

Limitations: The experts also offered several suggestions for improvements to the framework. E0 discussed the possibility of supporting comparisons between more than two models. *"This can speed up the fairness-oriented model selection procedure if a number of models can be compared and analyzed at once."*. E2 recommended that for groups in the rank mapping view, the details of the advantaged and disadvantaged nodes can be queried. For example, the analyst would like to highlight specific nodes in a group. E3 and E4 found the interface to be initially challenging, and these experts required the longest amount of time for training (10 minutes). They also often needed a reintroduction to views, and E5 commented that the framework has a relatively high learning curve for analysts who are not in the field of graph mining. E5 suggested adding information panels for each view may, and we have updated the system to incorporate this. Each view now contains a small question mark in the upper-right corner that provides a description of the view functionality on mouse-over.

Chapter 5

MACHINE LEARNING ROBUSTNESS ANALYSIS

So far, all of the previous chapters leverage visual analytics methods to explore and discover vulnerabilities in machine learning models and explain how these vulnerabilities may be occurring. However, a key question that comes up when a vulnerability is discovered is - now what? This chapter explores how we can suggest and highlight mechanisms for addressing machine learning vulnerabilities. Can the visual analytics system highlight example data elements that are under or oversampled and suggest potential mechanisms for reducing vulnerabilities? How can visual analytics best enable model developers to build and develop strategies to make the model more robust? How can we leverage those insights to motivate researchers to prevent the model from certain threats? Thus, the next step is to facilitate defense from the potential vulnerabilities of machine learning models with visual analytics approaches via human-in-the-loop analysis.

In this chapter, two new visual analytics frameworks have been introduced to enhance the accuracy and reliability of machine learning models. The first framework is focused on out-of-distribution detection based knowledge validation, which addresses the challenge of detecting model errors caused by novel or unfamiliar data that is not included in the training set. By providing visualizations that enable the detection of such errors, this framework allows for better model validation and improves the reliability of the model's predictions.

The second framework is designed for data augmentation diagnosis, which is an essential technique to overcome the problem of limited training data. This framework provides visualizations that help to diagnose and evaluate the effectiveness of data augmentation methods used to expand the training data. By identifying the strengths and weaknesses of different

Figure 22. Identifying and reasoning the known unknown instance to ResNet-34 trained on CIFAR-10. (A) The analyst selects Stage 0 as the base stage and Stage 1 as the current stage to investigate how well the model performs and what aspects need to be improved. From the Information Highlight View (B), the analyst notices that both training accuracy and testing accuracy has increased as 30,800 data instances are newly added to the training dataset, which can be verified from the Performance Linechart (C). The Diverging Misprediction Barchart (D) also indicates that the number of mispredicted instances reduced significantly, while the analyst also finds that the number of instances that are misclassified as "dog"s by the ResNet-34 is the largest among others (D.1). By filtering the Instance Gallery (E), the analyst further notices that the instance "cifar10_test_61" is misclassified as "dog" twice from both the base stage and current stage. To reason the instance "cifar10_test_61", the analyst clicks the corresponding row and discovers that the similar instances are from either "dog" or "cat" (F), and they are heavily overlapped on their softmax distribution similarity, which means the ResNet-34 currently has difficulty in distinguishing some "dog"s and "cat"s. Such a hypothesis can also be verified in the Instance Detail View as the selected instance "cifar10_test_61" is difficult to be distinguished by humans as well.

data augmentation techniques, this framework enables the selection of the most appropriate methods for specific machine learning tasks. Both of these frameworks are valuable tools for improving the performance and reliability of machine learning models, and they represent significant contributions to the field of visual analytics.

5.1 Machine Learning Knowledge Validation

Machine learning models are widely used in commercial systems in various applications (Covington, Adams, and Sargin 2016; Ronneberger, Fischer, and Brox 2015; Esteva et al. 2017). These systems constantly collect new training data and update models in order to match the up-to-date data distributions. However, the real-world data collection procedures are challenging due to the form and size of the data. The collected data may be biased and not able to cover all the possible varieties in the feature space. While a model may perform well on known data, the inherent unknowns in the real-world leave models open to various vulnerability issues, including poor performance on out-of-distribution data (Hendrycks and Gimpel 2018; Lee et al. 2018; Liang, Li, and Srikant 2020; C. Chen et al. 2020) and adversarial examples (goodfellow2014explaining; Huang et al. 2011; Vorobeychik and Kantarcioglu 2018b; Das et al. 2020; Brown et al. 2018; Shamsabadi, Sanchez-Matilla, and Cavallaro 2020; Zhao, Liu, and Larson 2020; Athalye et al. 2018). The former is when the model incorrectly predicts unlabeled data that is out of training data's distribution, while the latter is the case that the model is fooled by artificially-crafted data with imperceptible subtle perturbations. Essential to these vulnerability issues is that the models cannot handle unknown data, which requires analysts to understand and explore what has been learned in the black-box models. Although there are many existing works addressing the identified vulnerabilities (Das et al. 2018; Rahnama, Nguyen, and Raff 2020; Xiao and Zheng 2019), new vulnerabilities still arise as long as the models are not perfect and the training data is limited. In fact, it is almost impossible to train a perfect model to cover known and unknown situations in practice, leaving us a question: how do we validate whether a model is well-trained or not? The traditional statistical metrics such as accuracy and recall are not enough for revealing what is learned in a model and how well it learns. Take an image classifier as an example. 100% accuracy on training and testing datasets does not mean the model is perfect; instead, there are always some new images the model cannot predict correctly. Even if we train models that perform well on any known data, adversarial examples can be generated to confuse the model. Thus, the key question becomes how we validate the knowledge learned by a model, i.e., how do we align what we think the model is performing and what the model is actually learning.

We refer to the aforementioned challenges as *Machine Learning Model Knowledge Validation*, meaning the process of alignment of what knowledge the machine learning model has learned and what human believes the model has learned. For simplicity, we use the term *Knowledge Validation* to represent this concept throughout the paper.

In this paper, the main method of the knowledge validation is to infer the current deficiencies of the model by analyzing the causes of the mispredicting certain data instances and improve the model accordingly. For example, we infer a misclassified image as out-of-distribution data, meaning that the model is not well trained for predicting such an anomalous data instance. Correspondingly, some misclassified images are so confusing that even humans cannot recognize them, making the model mispredict them with high confidence. We categorize the former type of mispredicted data as *unknown unknowns*[7] and the latter as *known unknowns*[8] where adversarial samples belong to.

The knowledge validation is a dynamic process for distinguishing and reasoning those unknowns. It is required as long as the model accepts the new data and keeps evolving since such validation requires human cognition and involvement at every stage of the model, which also implies that the knowledge validation is difficult to quantify and achieve with automated algorithms. Therefore, we employ a human-in-the-loop approach to address this challenge. Our proposed visual analytics framework is developed under the model-agnostic setting, allowing analysts to perform instance-level inspections to facilitate prediction reasoning across each stage of the model development. The fully-adaptive components convey insights via both visualizations and text summarizations, which makes it easy to understand the model's behaviors and be guided for model improvement suggestions. Finally, we demonstrate the proposed framework with three usage scenarios to show the usability of the system for identifying and reasoning new unknown unknowns, existing known unknowns, as well

[7]An unknown unknown is also referred to as *Distribution Uncertainty* (Malinin and Gales 2018) which is caused by the mismatch between training and testing data distribution

[8]A known unknown is also referred as *Data Uncertainty* (Malinin and Gales 2018) caused by the complexity in the dataset, such as overlapping between different classes.

as adversarial examples generated by an attack algorithm FGSM (goodfellow2014explaining) through validating a well-known deep learning model ResNet (He et al. 2016) trained with an image dataset CIFAR-10 (Krizhevsky, Nair, and Hinton, n.d.). Contributions include:

- A visual analytics framework that supports analysts to perform machine learning model knowledge validation of any machine learning model.
- Interactive methods for progressively syncing model's instance-level prediction logic with domain experts.

From our literature review on machine learning knowledge validation and knowledge validation in visualization (C. Chen et al. 2020; Lee et al. 2018; Yuan et al. 2021), we have identified several research challenges and gaps in the literature. These challenges were then evaluated by four researchers in machine learning (two of which serve as co-authors on this paper). After iterative discussions with the experts, two major research challenges for machine learning knowledge validation were identified:

Interactive Incremental Model Improvement Support. As Yuan et al. (Yuan et al. 2021) mentioned, current visual analytics techniques for machine learning fall into three categories that provide support at different development stages: before machine learning model building, during the model building, as well as after the model building. However, from the incremental learning perspective, the model keeps evolving as long as there are new data to be trained. In this case, even a well-trained model is deployed as a commercial application will still suffer mispredicting the new data for many reasons, such as OoD data and adversarial examples that are generated by the new attack algorithms. Thus, for some tasks such as image classification, being deployed to real-world applications is not the end of the story. It is necessary to understand the model's behavior at every stage and obtain insights of what every misprediction delivers to us.

Instance-level Behavior Reasoning. Automatic strategies (Lee et al. 2018; Hendrycks and Gimpel 2018; Guo et al. 2017; Shalev, Adi, and Keshet 2019; Lee et al. 2017; Lakkaraju et al. 2016; Lakshminarayanan, Pritzel, and Blundell 2017) for mispredicted instances reasoning seem promising as they qualify them as either known unknown or unknown unknown. However, sometimes we need the granularity of knowledge validation can be detailed down to the instance level. Take image classification as an example, the model can have poor performance on both OoD data and adversarial examples, which are both treated as unknown unknowns. In this case, the analyst is not able to know the semantic context of those instances unless inspects and reasons the model's behavior on each one in detail, which can be different from case to case. Therefore, instance-level behavior reasoning is also necessary during knowledge validation.

5.1.1 Analytical Tasks

According to the aforementioned research challenges and gaps, we have identified three essential analytical tasks with our collaborators. These analytical tasks are refined iteratively during the framework development. In the end, we decided to employ image classification as our major machine learning task and summarized the following specific analytical requirements with regards to it.

5.1.1.1 T1: Organizing and Summarizing Model at Every Stage

Models of different stages should be summarized, which includes basic summarization of models' performance changes and dataset changes. In particular, the analysts should be able to know:

- *T1.1:* What is the performance of the model at each stage, and;
- *T1.2:* What is the dataset status at each stage.

5.1.1.2 T2: Identifying and Reasoning the Mispredicted Instances

Analysts should be able to identify the mispredicted instances and diagnose the reason for those mispredicted instances. As mentioned earlier, the mispredicted instances belong to one of the two categories: known unknown and unknown unknown. Thus, analysts want to understand:

- *T2.1:* What difficulty does the model currently have in predicting which classes?
- *T2.2:* What is the reason for a certain instance being misclassified? Does it belong to known unknown or unknown unknown? Why?

5.1.1.3 T3: Enabling Stage-wise Comparison and Validation

The analysts should be able to notice the difference of the model's behaviors through stage-wise comparison. In particular, analysts want to get the answers to the following questions:

- *T3.1:* Compared with previous stages, what are new issues in predicting new encountered data?
- *T3.2:* Compared with previous stages, are existing issues fixed? Has the model evolved by improving its prediction logic?

5.1.2 Design Requirements

Based on the analytical tasks, we engaged in an agile design process involving multiple iterations of the framework in collaboration with our domain experts. We have identified several design requirements and mapped different analytic tasks to each requirement.

5.1.2.1 D1: Summarize Model Performance with Highlights

The system should support the summarization of model performance (T1.1). Highlights of the model performance and data information at each stage should be extracted for simplicity (T1.2). In addition, the summarization should be standing out to guide analysts on what issues may need to be concerned for each stage, where the summarization of each stage is comparable (T3.1, T3.2) for validation with preferred reference.

5.1.2.2 D2: Summarize the Mispredicted Instances with Highlights

The system should visualize the overview of mispredicted instances (T2.1). Such an overview should also be comparable (T3.1, T3.2) and provide useful highlights for potential starting points of interest.

5.1.2.3 D3: Reason the Mispredicted Instance in Detail

The system should also be able to visualize detail of mispredicted instances (T2.2) for reasoning and model behavior understanding. The design detail view is also comparable across the model in different stages to provide evidence of the diagnosis of those mispredicted instances (T3.1, T3.2).

In summary, we specify the views that need to be developed and map their functionality to the analysis tasks that can be performed as follows:

- *Information Highlight View*, which integrates information including model performance, dataset changes, and misprediction information (T1.1, T1.2, T2.1) of selected stages (T3.1, T3.2).
- *Performance Linechart*, which compares the performance of the model across different stages (T1.1, T3.1, T3.2).
- *Diverging Misprediction Barchart*, which summarizes how the model performs in predicting each class of selected stages (T2.1, T3.1, T3.2).
- *Instance Gallery & Instance Detail View*, which shows mispredicted instances for detailed analysis (T2.2).
- *Instance Reasoning View*, which is used for reasoning how/why the mispredicted instance of interest is caused. (T2.2, T3.1, T3.2)

5.1.3 Visual Analytics Framework

Based on the analytic tasks and design requirements, we have developed a visual analytics framework (Figure 23) to support machine learning knowledge validation. The framework is designed to keep each model stage as a snapshot. The analyst can check out any two stages as the base stage and the current stage for diagnosis. The summarization is then automatically generated to highlight the potential issues from mispredicted instances (Figure 23.A). Then the analyst can interactively reason how and why certain instances of interest are mis-

Figure 23. The designed framework considers the model at different stages as a stage chain. It consists of two stages: (A) the stage summarization stage, and (B) the instance-level reasoning stage. In stage (A), the analyst can select the base stage and the current stage to be inspected. All necessary information such as model performance and dataset changes is retrieved and integrated for inspection. The combined summarization view can automatically highlight what is potentially interesting from mispredicted instances. In stage (B), the analyst can then explore and inspect instances of interest with the provided highlights. Through examing how and why certain instances are mispredicted, the analysts can obtain insights of the deficiency of the model and make improvements accordingly. The improved model can be viewed as a new stage for future diagnosis.

predicted and obtain insights of the deficiency of the model. (Figure 23.B). The analyst can leverage those insights to improve the model and record the improved model as a new stage for future diagnosis.

The framework supports two major functionalities: 1) stage summarization and 2) instance-level reasoning. The analysts are free to compare any two stages during the model development to explore how the model evolves through the Performance Linechart and the Diverging Misprediction Barchart. Our highlight algorithm can automatically provide recommendations for analysts to diagnose potential model deficiency. Analysts can then dive into the instance-level reasoning of the mispredicted instances of interest. The instance reasoning view provides both visual and textual explanation integrated with state-of-art OoD detection method (Lee et al. 2018) to show how the model sees the selected instance as. If it is a known unknown, it may suggest that the model has difficulty distinguishing the selected instance base on its knowledge. If it is an unknown unknown, it may suggest that the model lacks such data to be trained as it is some pattern the model has not seen before. Our modular design is model agnostic, which enables analysts to freely integrate any machine learning models, corresponding OoD detection methods, as well as classification tasks. For demonstration purposes, we employ image classification as our task.

5.1.3.1 Background of Related Machine Learning Techniques

ResNet (He et al. 2016) is a well-known deep neural network invented by He et al.. The ResNet model is designed with double- or triple-layer skips that contain nonlinearities and batch normalization in between to address the vanishing gradients problem as well as the degradation problem. The idea of ResNet is to build the network with the building block:

$$\mathbf{y} = \mathrm{F}(\mathbf{x}, \{W_i\}) + \mathbf{x} \tag{5.1}$$

Here $\mathbf{x}$ and $\mathbf{y}$ are the input and output vectors of the layers considered, and W_i is the weight matrix. The function $\mathrm{F}(\mathbf{x}, \{W_i\})$ denotes the residual mapping to be learned, and the operation $\mathrm{F} + \mathbf{x}$ consists of a shortcut connection and element-wise addition. We utilize the classical ResNet-34 model with 34 parameter layers as our major model to be validated for demonstration.

UniOoDD (Lee et al. 2018) is refereed as the unified framework for detecting both OoD data and adversarial examples proposed by Lee et al.. UniOoDD obtains the class conditional Gaussian distributions with respect to levels of features of the deep models under Gaussian discriminant analysis. The output represents a confidence score based on the Mahalanobis distance. In particular, the Mahalanobis distance-based confidence score $\mathrm{M}(\mathbf{x})$ is represented as:

$$\mathrm{M}(\mathbf{x}) = \max_c - (\mathbf{f}(\mathbf{x}) - \hat{\mu}_c)^{\mathsf{T}} \hat{\Sigma}^{-1} (\mathbf{f}(\mathbf{x}) - \hat{\mu}_c) \tag{5.2}$$

where $\hat{\mu_c}$ and $\hat{\Sigma}$ is the class mean and covariance of training samples:

$$\hat{\mu}_c = \frac{1}{N_c} \Sigma_{i:y_i=c} f(\mathbf{x}_i), \hat{\Sigma} = \frac{1}{N} \Sigma_c \Sigma_{i:y_i=c} (f(\mathbf{x}_i - \hat{\mu}_c))(f(\mathbf{x}_i) - \hat{\mu}_c))^{\mathsf{T}} \tag{5.3}$$

UniOoDD has been proven to be significantly effective on OoD detection as well as adversarial examples detection. We employ this method as an assisting algorithm to judge if a given mispredicted instance belongs to known unknown or unknown unknown, and encode the measured value of each instance into our visualizations.

FGSM (goodfellow2014explaining) is refereed as *Fast Gradient Sign Method* of generating adversarial examples proposed by Goodfellow et al.. The core idea of FGSM is to add fine-tuned noise to original data, and the perturbed data is shown as a wrong prediction with a high probability produced by the model. The added noise is represented as the following:

$$\eta = \epsilon \text{sign}(\nabla_{\mathbf{x}} J(\theta, \mathbf{x}, y)) \tag{5.4}$$

where θ be the parameters of a model, $\mathbf{x}$ the input to the model, y the targets associated with $\mathbf{x}$ and $J(\theta, \mathbf{x}, y)$ be the cost used to train the neural network. We use FGSM to generate adversarial examples in Section 5.1.4.3 to show how such instances can be identified and reasoned by our framework.

5.1.3.2 Summarize the Model Performance with Highlights

In the development of machine learning models, the goal is to have the model to perform as expected at every stage. Therefore, we want the system we design to help analysts quickly obtain information about the model's performance. We start with the simplest metrics used to measure the performance of the model. We design the *Performance Linechart* (Figure 22.C) to visually represent how the model performs at each stage (*T1.1*). We use different colors to represent different measures, and the Performance Linechart can be extended to support any numerical metrics, so that the analyst can choose the metrics he or she wants to examine by preference, such as accuracy, recall, etc. However, the line chart also has problems. Recently, Fan et al. (Fan et al., n.d.) mention that many factors of line charts can be confusing and misleading. Small changes in model performance in percentages are sometimes difficult to detect, and such changes can be large in reality. For example, if the test accuracy is reduced by 0.1%, but the size of the dataset is 1 million, then there are 1,000 mispredicted instances. Therefore, in addition to the customizable Performance Linechart, we design *Information Highlight View* (Figure 22.B) to record the changes in the model between the two model stages, including changes in the performance and the dataset *(T1.1 & T1.2 & T3.1 & T3.2)*. We use red [9] to indicate unfavorable information, such as worse performance, reduced dataset, etc., and green for favorable information, such as better performance and increased dataset. The analyst can obtain overall information on the model performance from the Information Highlight View and the Performance Linechart by checking out the information from the different stages.

5.1.3.3 Summarize the Mispredicted Instances with Highlights

We also want the system to help analysts quickly obtain information about the mispredicted instances. Such information can tell us the shortcomings of the model at the selected stage. Similarly, we use the *Diverging Misprediction Barcharts* (Figure 22.D) to show the number of data that the model incorrectly predicts in the base and the current stages (*T2.1 & T3.1 & T3.2*). For example in an image classifier, each bar of the Diverging Misprediction Barcharts indicates that the model misclassified the instance into a certain category, and each bar is stacked with training data portion and test data portion respectively. However, such a design is not scalable with respect to the number of classes. We use CIFAR-10 as the dataset for image classification, which has a total of 10 classes. However, real-world datasets can be much more diverse, such as CIFAR-100 and ImageNet (Deng et al. 2009) that have 100 cat-

[9]For color-blind friendly consideration, we have also provided another set of color encoding alternatives to improve the user experience.

egories or more. Imagine so much statistical information being displayed in the Diverging Misprediction Barchart, analysts can have difficulties in finding the information they need. Therefore, we design a simple but effective highlighting algorithm to select the important information from the Diverging Misprediction Barchart, mainly showing which classes the model mispredict have the most noticeable changes. To do so, the challenge is to employ what metric to represent the significance of changes. We can use the absolute difference to represent the change of certain classes between the two stages, but the largest absolute difference does not necessarily represent the most noteworthy change (e.g., 1,000 instances are misclassified as cats represent only 0.1% of all misclassified instances). Similarly, we can use the ratio of a difference to represent relative changes, but the largest change ratio does not necessarily represent the most noteworthy change as well (e.g., 1 new instance misclassified as a cat represents 50% of all instances misclassified as cats). Therefore, we employ the product of the absolute difference and the ratio of the difference as the significance of the change, which considers both absolute and relative changes. We sort each misclassification by the computed significance and select the top-5 of them to be displayed in the Information Highlight View. For example, the analysts will read that "the number of instances misclassified as dogs have increased by 30% compared to the previous stage". Such noteworthy changes help guide the analyst to examine specific categories rather than browsing the Diverging Misprediction Barchart without a goal. Finally, each portion of the bar chart is also clickable. Analysts can filter instances of error categories by clicking on the specified section. These instances will be displayed in the *Instance Gallery* (Figure 22.E).

In addition to getting the statistical information by looking at the Diverging Misprediction Barchart, we also design the Instance Gallery to display the mispredicted instances (*T2.2*). The Instance Gallery consists of two parts, the upper shows a summary of the mispredicted instances as well as the customizable filtering options, and the lower part shows the

Table 1. Observation and Interpretation of Instance Reasoning View

Description	Observation	Interpretation
The selected instance is unknown unknown	The selected instance is far from the similar instances horizontally	"I have never seen this image before."
The selected instance is known unknown	The selected instance is in between the similar instances horizontally	"I know this species because I was trained with similar ones, but this one is confusing."
The selected instance is more unique to the model	The selected instance is far from the similar instances vertically	"To my best knowledge, these are similar instances that help me predict, although they are subtly different from the given instance."
The selected instance is less unique to the model	The selected instance is close to similar instances vertically	"According to what I have learned, these instances share the similar features."

thumbnails of instances which are grouped by class. Because we want to diagnose the reason why the instances are misclassified, we use the UniOoDD framework (Lee et al. 2018) to help detect whether the misclassified instances are unknown unknown or known unknown. The UniOoDD outputs the confidence score for each instance that denotes how confident the model predicts the given instance. Since such a confidence score is multidimensional, the OoD detector can be achieved by training a classifier with the confidence score as the feature, and the ID/OoD as the label. We retrain such classifier at every stage as an auxiliary method to perform a cursory evaluation of the current misclassified instances, with the training data as the In-Distribution (ID) data and the dataset that is rather than CIFAR-10 as the OoD data (such as SVNH (Netzer et al. 2011)). We extract the proportion of unknown/known unknowns and summarize it with text on the top of the Instance Gallery. In addition, these two terms are colored in orange and yellow, and we use the same colors to fill the border of the thumbnails to denote which unknowns the instance belongs to. We also design six filtering options to help analysts filter the mispredicted instances, which include the instance label, the instance prediction is base/current stage, whether the instance is newly added, the instance is from the training/testing set, and the instance is known unknown or unknown unknown. Finally, all thumbnails support hovering to show the details with a tooltip.

5.1.3.4 Reason the Mispredicted Instance in Detail

Since we also want to reason the mispredicted instances in details, we further employ the *Instance Reasoning View* (Figure 22.F) to reason how and why the instance is diagnosed as known/unknown unknown (*T2.2*). As mentioned before, Chen et al. (C. Chen et al. 2020) consider misclassified instances as either known unknown or unknown unknown and show that different kinds of misclassified instances essentially embody different problems of the model. For example, the implication of an unknown unknown is that the model has low confidence in predicting such an instance. We elaborate further on the relationship between the properties of instances and model performance based on previous work. We propose that model performance is reflected in three aspects of the mispredicted instance: (1) the model's confidence in predicting the instance, (2) the evidence/knowledge the model uses for the prediction, and (3) the model's ability to apply the learned knowledge.

1. *Model's confidence predicting the instance.* The confidence of the model for the predicted instance reflects how well the model knows this instance. Hendrycks et al. (Hendrycks and Gimpel 2018) have demonstrated that the model's output at the softmax layer does not represent the model's confidence level as the model also misclassifies OoD data with high values. Instead, the confidence score proposed by Lee et al. (Lee et al. 2018) shows such a feature of the model. The confidence level of the model outputting a misclassified instance has different meanings. High confidence

but misclassified instances indicate that the model has learned similar instances but is not yet able to distinguish them well from other classes (known unknowns). The misclassified instances with low confidence indicate that the model is unfamiliar with such instances (unknown unknowns). We utilize UniOoDD as an assisting method to obtain the confidence score for each prediction. Instances marked as known unknown mean that the model predicts them with high confidence, while instances marked as unknown unknown show low confidence prediction.

2. *The evidence/knowledge of model prediction.* The evidence of model prediction represents the basis on which the model classifies an instance. Molnar (Molnar 2020) introduces an instance-based interpretation, meaning that the way the model judges this instance is based on the fact that the way the model also judges its similar instances. We apply this interpretation to speculate on the evidence of the model prediction as such an interpretation is easier to understand and more likely to respond to the local behavior of the model. Since the softmax layer is considered to be the knowledge learned by the model (Hinton, Vinyals, and Dean 2015), we consider that the instances considered similar by the model have similar knowledge structures, i.e., similar softmax distributions. We compute the KL divergence of the softmax distribution between instances to obtain the similarity of knowledge. Ideally, instances of the same class have similar softmax distribution in model understanding, and such a distribution makes them predicted to be in the same class. For the OoD data or adversarial examples, their softmax distribution is different from the ID data, and thus they are misclassified, although humans think they are perceptually similar to the category they should belong to.
3. *The model's ability to apply the learned knowledge.* Finally, the ability of the model to apply the knowledge indicates whether the model makes accurate judgments about

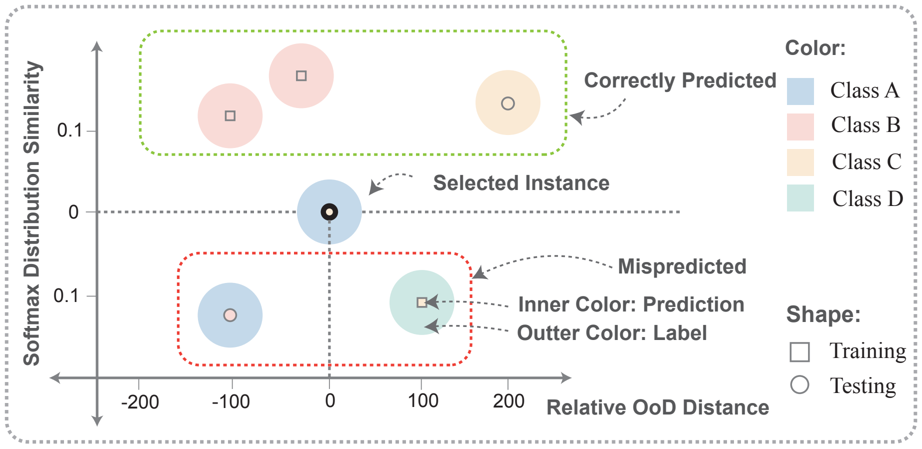


Figure 24. Visual encoding of the Instance Reasoning View. We encode the circles to be the instances from the testing data, and squares to be the instances from the training data. The color of circles or squares denotes the prediction class of the instance. In addition, each instance also contains a larger concentric circle. The color of this concentric circle represents the label of the instance itself. We also use the x-axis to represent the Relative OoD Distance of the model for the instances. Among other similar instances, the further the selected instance is from other similar instances, the more unfamiliar the model is to that instance. Finally, we use the y-axis to represent the softmax distribution similarity, the closer to zero means the instances are more similar to the selected instance in the model's understanding. The correctly predicted and the mispredicted instances are separated on the top and the bottom side respectively.

the acquired knowledge. Similar to the concept of underfitting, the low accuracy of the model for the training data is also the reason why the model predicts incorrectly for a certain instance. Therefore, when we obtain the instances that the model considers similar, it is critical that these instances are predicted correctly. If the model does not perform well in predicting the vast majority of similar instances, then the model is not likely to be well trained.

Instance Reasoning View. To better show the inference of incorrectly predicted instances for the model, we designed the Instance Reasoning view to show those aspects in detail (*T2.2*). As shown in Figure 24, we encode the instance information into a scatter plot. For a given model stage and the selected instance, we select 20 instances that are the most similar to the selected instances by examining the softmax distribution of all instances, among which 10 are from the training set and 10 are from the testing set. We encode each instance as a circle if it is from the testing set, or a square if it is from the training set. The color of circles or squares denotes the prediction class of the instance. In addition, each instance also contains a larger concentric circle. The color of this concentric circle represents the label of the instance itself. We use the y-axis to indicate the similarity between the selected and similar instances (i.e., the value of KL divergence between the two softmax distributions) such that a smaller distance vertically indicates a more similar of those two instances. For clarity, we also separate the instances into two portions, where instances above the selected instance mean that they are correctly classified, and those below the selected instance mean that they are incorrectly classified. We also encode the x-axis to represent whether the model is confident to predict the selected instance. As we mentioned before, Lee et al. (Lee et al. 2018) use the confidence score extracted from the framework as a feature of the instances, using whether it is OoD data as a label. we extract such confidence score of each instance as the encoding of the x-axis. When the confidence score is multidimensional, we apply PCA to obtain a numerical confidence score with the largest variance as the value of the x-axis for each instance, which here is called *Relative OoD Distance*. In this way, we can determine the confidence level of the model for a selected instance by observing the relative position of the levels of that instance and its similar instances. As listed in Table 1, if the selected instance is in the middle of the training set, then the model is familiar with the pattern of the current instance, but the model is not accurate enough to make a correct prediction (known unknown). If the selected instance is horizontally distant from the training set, the model is unknown to the current instance. We also encode the circles that represent the instances. The circle and the texture and color represent whether the instance belongs to the training set or the test set. In addition, we also

Figure 25. Identifying and reasoning the unknown unknown instance to ResNet-34 trained on CIFAR-10. The analyst selects Stage 0 as the base stage and Stage 1 as the current stage to investigate how well the model performs and what aspects need to be improved. From the Information Highlight View (A), the analyst notices that both training accuracy and testing accuracy has increased as 30,800 data instances are newly added to the training dataset, which is also shown in the Performance Linechart (A.1). The Diverging Misprediction Barchart (A.2) indicates that the number of mispredicted instances is reduced significantly. The analyst wonders what are the newly mispredicted instances. Through filtering the Instance Galley (B), the analyst further notices that the new instance "cifar10_train_33812" is misclassified as "cat" with the label "frog". By clicking the instance "cifar10_train_33812", the analyst discovers that the similar instances are from either "dog" or "cat" (C.2), and they are heavily overlapped on their softmax distribution similarity, which means the ResNet-34 currently has difficulty in distinguishing some "dog"s and "cat"s. Such a hypothesis can also be verified in the Instance Detail View as the selected instance "cifar10_train_33812" is difficult to be distinguished by humans.

made a textual summary of these three observations to explain the phenomena presented in the scatter plot. To facilitate the observation, we also design the *Instance Detail View* to show the actual images of all the instances that appear in the scatter plot. Finally, the system also dynamically adjusts to the stage in which the selected instance appears. If the instance is present in both the base and current stages, the view automatically displays two scatter plots representing two different stages. The analyst can observe the changes in the predictive logic of the instance in the model evolution by performing pairwise comparisons (*T3.1 & T3.2*).

5.1.4 Usage Scenarios

We present three usage scenarios to demonstrate how our framework supports knowledge validation during the model development process. We first show how analysts diagnose new unknown data of CIFAR-10 during evolving the ResNet-34 model, and then show how analysts diagnose mispredicted CIFAR-10 instances existing throughout the stages. Finally, we show the framework used for discovering and reasoning the adversarial examples of CIFAR-10 generated by FGSM.

5.1.4.1 Inspecting New Unknown Unknown

A common usage scenario is when a model encounters new and unknown data during its evolution. The model is usually unfamiliar with such data, but it can only be shown as a decrease in accuracy in the traditional performance measurements. Thus, the analyst needs to know how the model perceives such data and how to improve the model's performance based on what is reflected by such data. In this usage scenario, we simulate when the deep learning model ResNet-34 trained with CIFAR-10 encounters new unknown data and produces the wrong prediction on that data. We want to know what such wrong predictions tell us and how to improve. There are two stages in our simulation, where Stage 0 is the ResNet-34 that is trained with part of the CIFAR-10 training data (19,200 instances), and Stage 1 is when the ResNet-34 model is fully trained (50,000 instances). The purpose of this setting is to simulate the progress that the model is trained with new data gradually and keeps evolving.

Identifying the New Unknown Instances: After the simulation data is loaded, the analyst inspects the Information Highlight View to obtain the overall information of the model and data. As mentioned, the system automatically marks Stage 0 (not fully trained) as the base stage and Stage 1 (fully trained) as the current stage. The Information Highlight View shows that compared with the base stage, the current stage has increased the training accuracy by 20.110% (from 79.59% to 99.7%) and the testing accuracy of 14.580% (from 78.42% to 93.0%), which is reflected on the Performance Linechart (Figure 25.A.1). In terms of the size of the dataset, the current stage has grown the training dataset from 19,200 of the base stage to 50,000, and the size of the test dataset has no change. The Information Highlight View also indicates that the misprediction rate of almost all classes has dropped significantly. (Figure 25.A.2) The analyst then inspects the Diverging Misprediction Barchart to verify the summarizations, which shows that every class has dropped the misprediction rate significantly as bars of the current stage are much lower than those of the base stage. In addition, the analyst can hardly see the bars that represent mispredicted instances in the training dataset of the current stage, as the training accuracy is 99.7% nearly 100%. This shows the model is improving overall as the model is trained with more new data. The analyst then is wondering if there were newly introduced mispredicted instances that are OoD instances or unknown unknowns. By filtering the Instance Gallery (Figure 25.B), the analyst discovers that 2 instances are misclassified, one belongs to "frog" and one belongs to "ship" (Figure 25.B.1). The analyst finds that the first one ("cifar10_train_32812") is interesting as it is currently misclassified as "cat". It seems that the frog does not share very many common features with the cat, which makes the analyst curious about the reason for such misclassified instance (Figure 25.C).

Reason the New Unknown Instances: The analyst then wants to dive into instance "cifar10_train_32812" to reason how and why it is misclassified. By clicking the instance in the Instance Gallery, details are shown on both the Instance Reasoning View and the Instance Detail View. The former shows that the Relative OoD Distance between the selected instance "cifar10_train_32812" and similar instances are large (Figure 25.C.1), which indicates that the model is not confident to predict these instances compared with other similar instances. In addition, the analyst further notice that the model tries to align the learned instances of "cat" and "frog" to predict this instance, which indicates that the ResNet-34 model currently believes some instances of "cat" are similar to instances of "frog" (Figure 25.C.2). Finally, by looking at the Instance Detail View, the analyst finds the misclassified instance "cifar10_train_32812" is visually confusing to humans, and the frog's skin texture is also similar to some cats' skin texture, which is a possible reason for such prediction (Figure 25.C.3). As a result, the analyst can get the clue that *currently the model is not good at classifying such instances as not enough similar data is trained. Adding more similar data may help improve the model.*

5.1.4.2 Inspecting Known Unknown

The second usage scenario is when the model encounters instances that are mispredicted even after one or more stages of improvement. The data is not new to the model but still can not make correct predictions. Traditional metrics such as accuracy only show one aspect of performance and are inadequate for instance-level validation. In this case, the analyst needs to know what limited the model of recognition of such instances, and why the predictions are wrong. For example, if a picture of a cat is always classified by the model as a dog or a horse, it is likely that the model is not fine-grained enough to distinguish quadrupeds. So in this usage scenario, we continue to use the first usage scenario setting to explore those instances where the performance is not improved by increasing the size of the dataset.

Identifying the Known Unknown Instances: Since the simulated stages are the same as Section 5.1, we skip to the part when the analyst starts to look for the mispredicted instances of interest after the simulation data is loaded. Through browsing, the analyst finds that the model makes the mistake of misclassifying instances as "dog"s, and this misprediction rate is the highest compared to other categories (Figure 22.D.2). The analyst wonders what are the instances that model keeps mispredicting as "dog"s. The analyst then filters the Instance Gallery to select the instances that are mispredicted as the "dog" at both the base and the current stage. From the Instance Gallery view, the analyst discovers that 22 out of 36 (61.111%) instances tend to be unknown unknown, and 14 out of 36 (38.889%) instances tend to be known unknown (Figure 22.E). The analyst then finds in the table that the instance "cifar10_test_61" is labeled as "cat", however, the model keeps predicting it as "dog" even after the model is fully trained. The analyst then wonders what makes the model have difficulty in correctly classifying such an instance.

Reasoning the Known Unknown Instances: The analyst clicks on the instance "cifar10_test_61" in the Instance Gallery to find out the reason why it is misclassified as the "dog" twice. The Instance Reasoning View shows 2 scatter plots with corresponding summarizations. The former represents the base stage and the latter represents the current stage (Figure 22.F). The analyst finds that all similar instances of both scatter plots are horizontally aligned on a line that is close to y = 0, and these instances originate from either the "cat" or "dog" class. This indicates that the model considers the instance to be very similar to both the cat and dog classes at both stages. the position of the x-axis indicates that the selected instance is in the middle of similar instances. This indicates that the model is familiar with such instances at both stages, but the model is just not accurate enough to distinguish between the two categories of "cat"s and "dog"s. The analyst further observed Instance Detail View and found that the selected picture of a cat was also very confusing even to a human (Figure 22.G). It can be presumed that the features such as the nose that can distinguish between cats and dogs are blurry and the model can only take a guess. With this finding, the analyst can conclude that *the reason the model misclassified the instance is that the instance itself is confusing to be distinguished.*

Figure 26. Identifying and reasoning the adversarial examples encountered by ResNet-34 trained on CIFAR-10. The analyst selects Stage 1 as the base stage and Stage 2 as the current stage to investigate what aspects caused the testing accuracy to decrease (A). From the Information Highlight View (B), the analyst notices that the testing accuracy of Stage 2 has dropped 0.921%, and the warning section shows that model has increased the misprediction rate to incorrectly classify instances into classes such as "bird", "deer", etc. (C) The analyst now wonders what are instances that are misclassified as "deer". So the analyst filters the Instance Gallery and finds that the new instance "cifar10_test_10002" is misclassified as "deer" with its original label "airplane" (D). To reason the instance "cifar10_test_10002", the analyst discovers in the Instance Reasoning View that the similar instances in the training set are far from the selected instance both horizontally and vertically "cifar10_test_10002" (E), which means the ResNet-34 currently consider the instance is more unfamiliar than any learned instances. In the meanwhile, there are other instances that the model cannot correctly classify, all of which are from the test dataset (F). This indicates that the quality of the new test data needs to be reviewed.

5.1.4.3 Inspecting Adversarial Examples

The third usage scenario is when the model encounters adversarial examples. As mentioned earlier, models are vulnerable to adversarial attacks (Huang et al. 2011). The corresponding adversarial examples are generated to avoid being correctly predicted by those vulnerable models. In this case, it is critical to identify such adversarial examples. In this usage scenario, we add an additional stage base on the simulation used in Section 5.1.4.1 and Section 5.1.4.2. We add 100 adversarial examples generated by FGSM in the test set to simulate the scenario when the model encounters real-world data that is crafted by attackers with malicious purposes. The goal is to identify and reason those adversarial examples so that the machine learning researchers and developers can improve accordingly.

Identifying the Adversarial Examples: The analyst starts by setting Stage 1 as the base stage and Stage 2 as the current stage to investigate the performance changes (Figure 26.A). Through the Information Highlights View, the analyst finds that the current stage has its testing accuracy dropped 0.921% (from 93% to 92.079%), as the testing data size increases

100 (Figure 26.B). Automatically-generated warnings show that the misprediction rate that the model mispredicts instances as "deer" is 48.24%, which is the highest among others (Figure 26.C), making the analyst curious about what instances are misclassified by the model. Through filtering the Instance Gallery, The analyst also notices that the instance "cifar10_test_10002" is interestingly classified as "deer" instead of "airplane", and is diagnosed as unknown unknown. Since the "deer" and the "airplane" are very visually different, the analyst is curious how model performs such prediction, thus wants to investigate the instance "cifar10_test_10002" in detail (Figure26.D).

Reasoning the Adversarial Examples: As the analyst clicks the instance "cifar10_test_10002", the Instance Reasoning View shows the reason why the instance is diagnosed as unknown unknown. The analyst noticed that the selected instance "cifar10_test_10002" is horizontally far from the training instances that the model believes are similar, which indicates that the model has low confidence in predicting this instance as it has never seen such image before. In addition, the analyst also notices that the most similar instances (labeled as "deer") of instance "cifar10_test_10002" are vertically far from it, showing the model can hardly leverage the knowledge to predict the instance (Figure 26.E). On the contrary, there are also instances that the model believes to be similar to the instance "cifar10_test_10002", but almost all of them are mispredicted and belong to different classes, which indicates the model is already confused by the selected instance "cifar10_test_10002". Finally, the analyst finds in the Instance Detail View that the model has the same confusion about those instances and predicts all of them as "deer" as well (Figure 26.F). Thus the analyst can conclude that *the selected instance is unfamiliar to the model of the current stage, which can be treated as OoD data instance. One way to improve the model is to add* *similar instances to the training dataset for training.* [10] *In addition, the quality of the new test data needs to be reviewed as the model performs poorly on it.*

5.1.5 Evaluation and Expert Review

To further evaluate our framework, we conducted an interview with our collaborators (E0, E1), 4 domain experts (E2, E3, E4, E5) in machine learning and artificial intelligence. For the interview, we first introduced our system by describing the analytical tasks supported in the framework. We then demonstrate the components of our framework by walking through all three usage scenarios described in Section 5.1.4.1, Section 5.1.4.2 and 5.1.4.3. Finally, the experts were allowed to freely explore and inspect different instances and stages. The duration of the interview was approximately 90 minutes. On average, experts spent approximately 10 minutes learning and mastering the system and were able to explore different instances of inspection under the different stages. All experts were able to fully understand the major functionalities of the system by asking a few questions during the exploration phase. After the free exploration stage was finished, we collected feedback with numeric ratings from the experts using the following questions:

1. How effective is this framework in supporting model development? (*Q1.*)
2. How well our design supports the proposed analytical tasks? (*Q2.*)
3. Compared with other approaches, what are the pros and cons of the visual analytics approach to address the knowledge validation problem? (*Q3.*)
4. How would the framework fit into your development circle? (*Q4.*)

[10] Training with adversarial examples is also a simple yet effective approach of post-defense, also known as adversarial training, or retraining. (Vorobeychik and Kantarcioglu 2018b)

5. Please rate the user experience from 1 to 5 (poor to good) considering the intuitiveness and effectiveness of the framework. (*Q5.*)

Framework: We received mostly positive feedback on the overall design of the framework. The experts agreed that it is necessary and useful to have such a framework to help the machine learning knowledge validation, as it helps to build the model for the sake of the long term evolution. E0 and E1 considered that the design of keep tracking the model development with stages is similar to version control software such as git (Q1). E1 appreciated the design of the Instance Reasoning View, especially the scatter plot of the selected instance. *"I personally like the design of the Instance Reasoning View, as it visualizes the difference between the selected OoD data and ID data. The metric for OoD detection (UniOoDD) already shows a good performance of telling whether a given instance is OoD or not, but this view further explains why this is the case, which is more helpful for improving the model. (Q2, Q3) "* E2 and E5 also appreciated that the framework fits the general process of model development since most of the work focuses on well training a model with a given dataset, such as MNIST, SVHN, CIFAR-10, and ImageNet. However, the real-world data is not limited to those fixed-sized datasets, and it has to be learned incrementally. Having such a tool is helpful to address the issues. (Q4)

Visualizations: The experts all agreed on the effectiveness of the visualization design in our framework (Q1). They noted that all visualization components are simple but intuitive. E3 noted that the learning curve of this system is relatively low since there are less complicated visual designs in the system which facilitates understanding of the model. E4 appreciated the design of the Instance Reasoning View that can represent instance prediction/label, training/testing, and OoD Index and softmax distribution similarity at the same time. *"This view could assist us in checking how well the model learns in the current stage easily. The text* *description also helps us to gain insights in a narrative fashion."* The average score for the user experience question is 4.1667 $= \frac{3.5x2+5x2+4x2}{6}$ (Q5). All of our domain experts felt that the interactions were appropriate and provided the necessary information.

Improving Suggestions: The experts also offered several suggestions for improvements to the framework. E0 discussed the possibility of supporting model comparisons between more than two stages. *"It would be more interesting to check a selected instance's prediction changes over time by looking at multiple scatter plots, each for one stage."*. E4 recommended that the selected instances can be customized, given the current threshold of 20. For example, the analyst would like to update the threshold to see more related instances on the scatter plot of the Instance Reasoning View. E3 and E0 found some components to be initially challenging, for example, the Diverging Misprediction Barchart. They needed a reintroduction to this view that the bar represents the model classifies the instances as certain classes, instead of instances' original classes. E4 suggested linking the instances of the Instance Reasoning View and Instance Detail View may help match instances with actual pictures, and we have added this feature accordingly.

5.2 Data Augmentation Diagnosis

The detection of out-of-distribution samples in machine learning plays a crucial role in ensuring model reliability and generalization. Data augmentation, on the other hand, is a widely used technique to increase the diversity of training data and improve the performance of machine learning models. Interestingly, these two topics are interconnected in the context of addressing out-of-distribution detection challenges. By incorporating data augmentation techniques during the training phase, models are exposed to a broader range of variations, making them more capable of distinguishing between in-distribution and out-of-distribution samples. Data augmentation introduces synthetic examples that expand the representation space, enabling models to learn more robust and discriminative features. Consequently, this augmentation enhances the ability of models to identify and reject out-of-distribution samples by reducing the reliance on distribution-specific patterns. Thus, data augmentation serves as a valuable tool in strengthening the out-of-distribution detection capabilities of machine learning models.

The success of deep learning in natural language processing (NLP) (Devlin et al. 2018; Yinhan Liu et al. 2019; Vaswani et al. 2017) and computer vision (CV) (Krizhevsky, Sutskever, and Hinton 2017; Simonyan and Zisserman 2014; He et al. 2016) is often attributed to the increased amount of high-quality data (Deng et al. 2009; A. Wang et al. 2018). Data augmentation, a method to generate synthetic data from existing data, has gained attention, particularly in image augmentation (Jung et al. 2020; Hongyi Zhang et al. 2017; DeVries and Taylor 2017; Hendrycks et al. 2019; Cubuk et al. 2018; X. Chen et al. 2020; Chen, Kornblith, Norouzi, et al. 2020; Chen, Kornblith, Swersky, et al. 2020; Yang et al. 2020; Erichson et al. 2022; Lim et al. 2021). Besides addressing data scarcity, data augmentation has regularization effects that improve model quality (Hongyi Zhang et al. 2017; Hernández-García and

König 2018). Like other training and regularization techniques, such as Dropout (Srivastava et al. 2014) and BatchNorm (Ioffe and Szegedy 2015), data augmentation provides another parameter for practitioners to tune. However, selecting the optimal data augmentation or combination can be challenging due to the vast search space of available techniques. In the image domain alone, there are over twenty categories of data manipulations that can be combined in various strategies (Jung et al. 2020; Cubuk et al. 2018), making it difficult to identify the best approach.

Data science engineers face the challenge of finding the optimal augmentation among numerous approaches when dealing with data scarcity (Hendrycks et al. 2019; Cubuk et al. 2018). Standard solutions like grid search or random search (Bergstra and Bengio 2012) rely on accuracy tested on separate validation data. However, a lack of guidance for selecting correct hyperparameters and making appropriate assumptions about augmentation methods can result in significant resource costs in terms of time and computation. It is also important to avoid excessive "peeking" at validation data (Martin, Peng, and Mahoney 2021) when adjusting multiple hyperparameters, as is common when exploring data augmentation. There is a growing need for interpretable tools that offer guidance on improving models, such as data augmentation, particularly in applications that require more than just analyzing training, validation, and testing curves.

We present a visual analysis framework *AugLens*, which facilitates the process of explaining, diagnosing, and validating[11] deep learning models trained with data augmentation. We address the challenge of interpreting data augmentation by connecting supplemental data[12]

[11] We refer to the process of visually checking a trained model as "validating". Validation more commonly refers to a procedure for model selection in statistical learning, which should be differentiated from our concept of validating here.

[12] The term *supplemental data* denotes the synthetic data for data augmentation, which includes, but is not limited to, synthetic corrupted data (hendrycks2019benchmarking).

used in augmentation with the core property of neural networks, the loss function, visualized through loss landscapes. To visualize high-dimensional loss functions, we display loss landscapes (Yao et al. 2020; Yang et al. 2021) in low dimensions and its extracted structure in high dimensions.

We study data augmentation in detail at three scales, corresponding to the designed views in *AugLens* (Table 2): (1) *local-model scale*, including local model evaluations such as calculating leading Hessian eigenvalues (Yao et al. 2020), (2) *two-model scale*, featuring views of the CKA similarity (Kornblith et al. 2019) of a pair of selected models in both model-wise and layer-wise similarity, as well as loss landscape analysis using topological data analysis tools (Cohen-Steiner, Edelsbrunner, and Harer 2005; Edelsbrunner and Harer 2022), and (3) *multi-model scale*, which includes views of visualizing model parameter distribution and prediction distributions of local loss minimum ensembles.

The first two scales are motivated by a recent paper (Yang et al. 2021) analyzing deep learning model quality using loss landscape metrics. The third scale is a novel aspect of our system, designed to complement the first two. Each scale offers interpretable measurements, helping analysts gain insights during the image augmentation selection process. To demonstrate and validate our framework, we employ three representative case studies: MNIST augmentation on a multilayer perceptron (MLP) model (LeCun, Bengio, and Hinton 2015) with corrupted MNIST (MNIST-C (Mu and Gilmer 2019)), CIFAR-10 augmentation on ResNet-18 (DBLP:journals/corr/HeZRS15) with corrupted CIFAR-10 (CIFAR-10C (hendrycks2019benchmarking)), and CIFAR-10 augmentation on Vision Transformer (Dosovitskiy et al. 2020) with corrupted CIFAR-10C. Our contributions include:

- A multi-scale visual analytics framework, *AugLens*, measuring a model at the local-model scale, the two-model scale, and the multi-model scale;

Figure 27. Illustration of using our AugLens visual analytics system to diagnose the image augmentation of a two-layer perceptron model trained on MNIST and MNIST-C. (A) Browsing through the augmentation options, the analysts assume that the impact of an augmentation approach called "scale", i.e., adding scaled images, is minimal because simple scaling does not change content or clarity of the images. Both the model ensemble view (B) and the model similarity view (C) show that the un-augmented model "original" is similar to the augmented model "scale". However, the analysts observe significant performance differences in both the layer similarity view (D) and the evaluation view (E, F). The loss landscape view (G, H, I, J) shows that the model "scale" "lost" the minimum area compared to the model "original" on the validation dataset, which indicates that such augmentation does not improve the model. In fact, it results in worse performance of the model. The model parameter distribution view (K) further verifies that these two minimum areas do not have an intersection, and the model "scale" does not make consistent predictions on scaled images as evident in the prediction distributions view (L).

- A novel method for visualizing high-dimensional loss landscapes using topological data analysis, incorporating persistence barcodes (Cohen-Steiner, Edelsbrunner, and Harer 2005) and merge tree visualizations(Edelsbrunner and Harer 2022);
- Three case studies showcasing the application of *AugLens* in data augmentation for computer vision models with varying architectural complexities.

Our framework employs a modular design, allowing each module to be replaced according to analysts' needs. While the visualization techniques, such as those inspired by loss landscape analytics (Yang et al. 2021), are applied to image data augmentation tasks, they can easily be extended to support other tasks like model selection and model training. The data augmentation problem serves as a "case study" to bridge the gap between the visualization and machine learning communities. We expect our visualization system to be general enough for other model selection and training purposes when analyzing only training, validation, and testing curves is insufficient.

Table 2. AugLens analytical scales and their corresponding views.

Scales	*Tasks*	*Views*
Local-model	Performance-based Evaluation	Evaluation View (Performance, Fig. 27.E)
	Hessian-based Evaluation	Evaluation View (Hessian, Fig. 27.F)
Two-model	Loss Landscape Analysis	3D Loss Landscape (Fig. 27.G)
		Loss Heatmap (Fig. 27.H)
		Persistence Barcode (Fig. 27.I)
		Merge Tree (Fig. 27.J)
	Model-wise CKA Similarity	Model Ensemble & Similarity View (Fig. 27.B & C)
	Layer-wise CKA Similarity	Layer Similarity Matrix (Fig. 27.I)
Multi-model	Parameter Distribution Analysis	Parameter Distribution View (Fig. 27.K)
	Prediction Distribution Analysis	Prediction Distribution View (Fig. 27.L)

Chapter 6

CONCLUSION

In this work, five visual analytics frameworks are proposed to enhance the performance and reliability of machine learning models. These frameworks are designed to address different challenges in machine learning and provide valuable insights into the vulnerabilities of these models. By utilizing advanced visualization techniques, these frameworks enable users to identify potential vulnerabilities in machine learning models, as well as to evaluate the effectiveness of different augmentation techniques.

The first framework is for adversarial machine learning analysis, which enables users to identify and mitigate the potential vulnerabilities of machine learning models to adversarial attacks. This framework provides visualizations that allow users to detect the presence of adversarial attacks and to evaluate the robustness of machine learning models to these attacks.

The second framework focuses on sensitivity auditing of graph mining models, which is critical in identifying and mitigating the potential biases that may exist in the model. This framework enables users to visualize the sensitivity of the model's predictions to changes in the input data, allowing them to detect and correct biases that may be present in the model.

The third framework is for fairness analysis of graph mining models, which is an emerging area of research that seeks to understand and prevent algorithmic bias and discrimination in machine learning models. This framework provides visualizations that enable users to evaluate the fairness of graph mining models by examining the impact of sensitive attributes on the model's output. By integrating this framework into the development process of graph mining models, we can ensure that the models are fair and unbiased, which is crucial for making ethical and equitable decisions.

The fourth framework is for out-of-distribution detection based knowledge validation, which addresses the challenge of detecting model errors caused by novel or unfamiliar data that is not included in the training set. This framework provides visualizations that enable the detection of such errors, allowing for better model validation and improving the reliability of the model's predictions.

The fifth framework is designed for data augmentation diagnosis, which is an essential technique to overcome the problem of limited training data. This framework provides visualizations that help to diagnose and evaluate the effectiveness of data augmentation methods used to expand the training data. By identifying the strengths and weaknesses of different data augmentation techniques, this framework enables the selection of the most appropriate methods for specific machine learning tasks.

In conclusion, the proposed visual analytics frameworks demonstrate the effectiveness of these techniques in identifying vulnerabilities of machine learning models and improving the models based on different types of augmentation techniques. These frameworks have significant implications for the development of more reliable and robust machine learning models and represent important contributions to the field of visual analytics.

REFERENCES

Abadi, Martín, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. "Tensorflow: Large-scale machine learning on heterogeneous distributed systems." *arXiv preprint arXiv:1603.04467.*

Abdelzad, Vahdat, Krzysztof Czarnecki, Rick Salay, Taylor Denounden, Sachin Vernekar, and Buu Phan. 2019. *Detecting Out-of-Distribution Inputs in Deep Neural Networks Using an Early-Layer Output.* arXiv: 1910.10307 [cs.LG].

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

Ahn, Y., and Y. -R. Lin. 2020. "FairSight: Visual Analytics for Fairness in Decision Making." *IEEE Transactions on Visualization and Computer Graphics* 26 (1): 1086–1095. https://doi.org/10.1109/TVCG.2019.2934262.

Alsallakh, Bilal, Allan Hanbury, Helwig Hauser, Silvia Miksch, and Andreas Rauber. 2014. "Visual methods for analyzing probabilistic classification data." *IEEE Transactions on Visualization and Computer Graphics* 20 (12): 1703–1712.

Antoniou, Antreas, Amos Storkey, and Harrison Edwards. 2017. "Data augmentation generative adversarial networks." *arXiv preprint arXiv:1711.04340,* https://doi.org/https://doi.org/10.48550/arXiv.1711.04340.

Athalye, Anish, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. 2018. "Synthesizing Robust Adversarial Examples." In *Proceedings of the 35th International Conference on Machine Learning,* edited by Jennifer Dy and Andreas Krause, 80:284–293. Proceedings of Machine Learning Research. PMLR, October. http://proceedings.mlr.press/v80/athalye18b.html.

Barreno, Marco, Blaine Nelson, Anthony D. Joseph, and J. D. Tygar. 2010. "The security of machine learning." *Machine Learning* 81, no. 2 (November): 121–148.

Bergstra, James, and Yoshua Bengio. 2012. "Random search for hyper-parameter optimization." *Journal of machine learning research,* no. 2, https://doi.org/https://dl.acm.org/doi/10.5555/2188385.2188395.

Bertini, Enrico, and Denis Lalanne. 2009. "Surveying the complementary role of automatic data analysis and visualization in knowledge discovery." In *Proceedings of the ACM*

SIGKDD Workshop on Visual Analytics and Knowledge Discovery Integrating Automated Analysis with Interactive Exploration, 12–20.

Biggio, Battista, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. 2013. "Evasion Attacks Against Machine Learning at Test Time." In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases,* 387–402. Springer-Verlag.

Biggio, Battista, Giorgio Fumera, and Fabio Roli. 2014. "Security evaluation of pattern classifiers under attack." *IEEE Transactions on Knowledge and Data Engineering* 26 (4): 984–996.

Biggio, Battista, Giorgio Fumera, Paolo Russu, Luca Didaci, and Fabio Roli. 2015. "Adversarial biometric recognition: A review on biometric system security from the adversarial machine-learning perspective." *IEEE Signal Processing Magazine* 32 (5): 31–41.

Biggio, Battista, Blaine Nelson, and Pavel Laskov. 2012. "Poisoning Attacks Against Support Vector Machines." In *Proceedings of the International Conference on Machine Learning,* 1467–1474. Edinburgh, Scotland. http://dl.acm.org/citation.cfm?id=3042573.3042761.

Biggio, Battista, and Fabio Roli. 2018. "Wild patterns: Ten years after the rise of adversarial machine learning." *Pattern Recognition* 84:317–331.

Binns, Reuben. 2020. "On the Apparent Conflict between Individual and Group Fairness." In *Proceedings of the Conference on Fairness, Accountability, and Transparency,* 514–524.

Blanzieri, Enrico, and Anton Bryl. 2008. "A survey of learning-based techniques of email spam filtering." *Artificial Intelligence Review* 29 (1): 63–92.

Bose, Avishek, and William Hamilton. 2019. "Compositional Fairness Constraints for Graph Embeddings." In *Proceedings of the International Conference on Machine Learning,* 97:715–724.

Brown, Tom B., Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. 2018. *Adversarial Patch.* arXiv: 1712.09665 `[cs.CV]`.

Burkard, Cody, and Brent Lagesse. 2017. "Analysis of causative attacks against SVMs learning from data streams." In *Proceedings of the 3rd ACM on International Workshop on Security And Privacy Analytics,* 31–36. ACM.

Cabrera, Á. A., W. Epperson, F. Hohman, M. Kahng, J. Morgenstern, and D. H. Chau. 2019. "FAIRVIS: Visual Analytics for Discovering Intersectional Bias in Machine Learning." In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology,* 46–56. IEEE. https://doi.org/https://doi.org/10.48550/arXiv.1904.05419.

Cao, Kelei, Mengchen Liu, Hang Su, Jing Wu, Jun Zhu, and Shixia Liu. 2020. "Analyzing the noise robustness of deep neural networks." *IEEE transactions on visualization and computer graphics* 27 (7): 3289–3304.

Caruana, Godwin, and Maozhen Li. 2012. "A survey of emerging approaches to spam filtering." *ACM Computing Surveys* 44 (2): 9.

Cavallo, M., and C. Demiralp. 2019. "Clustrophile 2: Guided Visual Clustering Analysis." *IEEE Transactions on Visualization and Computer Graphics* 25 (1): 267–276.

Chartier, Timothy P, Erich Kreutzer, Amy N Langville, and Kathryn E Pedings. 2011. "Sensitivity and Stability of Ranking Vectors." *SIAM Journal on Scientific Computing* 33 (3): 1077–1102. https://doi.org/10.1137/090772745.

Chen, Changjian, Jun Yuan, Yafeng Lu, Yang Liu, Hang Su, Songtao Yuan, and Shixia Liu. 2020. *OoDAnalyzer: Interactive Analysis of Out-of-Distribution Samples.* arXiv: 2002.03103 [cs.HC].

Chen, Shang-Tse, Cory Cornelius, Jason Martin, and Duen Horng Polo Chau. 2018. "ShapeShifter: Robust Physical Adversarial Attack on Faster R-CNN Object Detector." In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases,* 52–68. Springer.

Chen, Ting, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. "A simple framework for contrastive learning of visual representations." In *Proceedings of the 37th International Conference on Machine Learning,* 1597–1607. Proceedings of Machine Learning Research, July. https://doi.org/https://doi.org/10.48550/arXiv.2002.05709.

Chen, Ting, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E Hinton. 2020. "Big self-supervised models are strong semi-supervised learners." *Advances in neural information processing systems,* 22243–22255. https://doi.org/https://doi.org/10.48550/arXiv.2006.10029.

Chen, Xinlei, Haoqi Fan, Ross Girshick, and Kaiming He. 2020. "Improved baselines with momentum contrastive learning." *arXiv preprint arXiv:2003.04297,* https://doi.org/https://doi.org/10.48550/arXiv.2003.04297.

Cohen, Jeremy, Elan Rosenfeld, and Zico Kolter. 2019. "Certified adversarial robustness via randomized smoothing." In *international conference on machine learning,* 1310–1320. PMLR. https://doi.org/https://doi.org/10.48550/arXiv.1902.02918.

Cohen-Steiner, David, Herbert Edelsbrunner, and John Harer. 2005. "Stability of persistence diagrams." In *Proceedings of the twenty-first annual symposium on Computational geometry,* 263–271. https://doi.org/https://doi.org/10.1007/s00454-006-1276-5.

Covington, Paul, Jay Adams, and Emre Sargin. 2016. "Deep Neural Networks for YouTube Recommendations." In *Proceedings of the 10th ACM Conference on Recommender Systems,* 191–198. RecSys '16. Boston, Massachusetts, USA: Association for Computing Machinery. https://doi-org.ezproxy1.lib.asu.edu/10.1145/2959100.2959190.

Cubuk, Ekin D, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. 2018. "AutoAugment: Learning augmentation policies from data." *arXiv preprint arXiv:1805.09501,* https://doi.org/https://doi.org/10.48550/arXiv.1805.09501.

Cubuk, Ekin D, Barret Zoph, Jonathon Shlens, and Quoc V Le. 2020. "RandAugment: Practical automated data augmentation with a reduced search space." In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops,* 702–703. https://doi.org/https://doi.org/10.48550/arXiv.1909.13719.

Dalvi, Nilesh, Pedro Domingos, Mausam, Sumit Sanghai, and Deepak Verma. 2004. "Adversarial classification." In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* 99–108. ACM.

Das, Nilaksh, Haekyu Park, Zijie J. Wang, Fred Hohman, Robert Firstman, Emily Rogers, and Duen Horng Chau. 2020. *Bluff: Interactively Deciphering Adversarial Attacks on Deep Neural Networks.* arXiv: 2009.02608 `[cs.LG]`.

Das, Nilaksh, Madhuri Shanbhogue, Shang-Tse Chen, Fred Hohman, Siwei Li, Li Chen, Michael E. Kounavis, and Duen Horng Chau. 2018. *Shield: Fast, Practical Defense and Vaccination for Deep Learning using JPEG Compression.* arXiv: 1802.06816 `[cs.CV]`.

Deng, Jia, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. "Imagenet: A large-scale hierarchical image database." In *2009 IEEE conference on computer vision and pattern recognition,* 248–255. Ieee. https://doi.org/10.1109/CVPR.2009.5206848.

Denouden, Taylor, Rick Salay, Krzysztof Czarnecki, Vahdat Abdelzad, Buu Phan, and Sachin Vernekar. 2018. *Improving Reconstruction Autoencoder Out-of-distribution Detection with Mahalanobis Distance.* arXiv: 1812.02765 `[cs.LG]`.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. "BERT: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805,* https://doi.org/https://doi.org/10.48550/arXiv.1810.04805.

DeVries, Terrance, and Graham W Taylor. 2017. "Improved regularization of convolutional neural networks with cutout." *arXiv preprint arXiv:1708.04552,* https://doi.org/https://doi.org/10.48550/arXiv.1708.04552.

Dosovitskiy, Alexey, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, et al. 2020. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale." *Clinical Orthopaedics and Related Research,* https://doi.org/https://doi.org/10.48550/arXiv.2010.11929. arXiv: 2010.11929.

Dua, Dheeru, and Casey Graff. 2017. *UCI Machine Learning Repository.* University of California, Irvine, School of Information and Computer Sciences. http://archive.ics.uci.edu/ml.

Dwork, Cynthia, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. 2012. "Fairness through Awareness." In *Proceedings of the Innovations in Theoretical Computer Science Conference,* 214–226.

Edelsbrunner, Herbert, and John L Harer. 2022. *Computational topology: an introduction.* American Mathematical Society.

Endert, A., W. Ribarsky, C. Turkay, B. L William Wong, I. Nabney, I. D??az Blanco, and F. Rossi. 2017. "The State of the Art in Integrating Machine Learning into Visual Analytics." *Computer Graphics Forum* 36 (8): 1–28.

Erichson, N Benjamin, Soon Hoe Lim, Francisco Utrera, Winnie Xu, Ziang Cao, and Michael W Mahoney. 2022. "NoisyMix: boosting robustness by combining data augmentations, stability training, and noise injections." *arXiv preprint arXiv:2202.01263,* https://doi.org/https://doi.org/10.48550/arXiv.2202.01263.

Esteva, Andre, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau, and Sebastian Thrun. 2017. "Dermatologist-level classification of skin cancer with deep neural networks." *nature* 542 (7639): 115–118.

Fan, Arlen, Yuxin Ma, Michelle Mancenido, and Ross Maciejewski. n.d. *Annotating Line Charts for Addressing Deception.* Association for Computing Machinery. https://doi.org/10.1145/3491102.3502138.

Friedler, Sorelle A., Carlos Scheidegger, and Suresh Venkatasubramanian. 2021. "The (Im)Possibility of Fairness: Different Value Systems Require Different Mechanisms for Fair Decision Making." *Communications of the ACM* 64 (4): 136–143.

Gatys, Leon A, Alexander S Ecker, and Matthias Bethge. 2015. "A neural algorithm of artistic style." *arXiv preprint arXiv:1508.06576,* https://doi.org/https://doi.org/10.48550/arXiv.1508.06576.

Gleich, David F. 2015. "PageRank Beyond the Web." *Society for Industrial and Applied Mathematics Review* 57 (3): 321–363.

Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2020. "Generative adversarial networks." *Communications of the ACM,* no. 11, 139–144. https://doi.org/https://dl.acm.org/doi/abs/10.1145/3422622.

Goodfellow, Ian, Jonathon Shlens, and Christian Szegedy. 2015. "Explaining and Harnessing Adversarial Examples." In *Proceedings of the International Conference on Learning Representations.* https://doi.org/https://doi.org/10.48550/arXiv.1412.6572.

Guo, Chuan, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. "On Calibration of Modern Neural Networks." In *Proceedings of the 34th International Conference on Machine Learning,* edited by Doina Precup and Yee Whye Teh, 70:1321–1330. Proceedings of Machine Learning Research. PMLR, June. https://proceedings.mlr.press/v70/guo17a.html.

Hardt, Moritz, Eric Price, and Nathan Srebro. 2016. "Equality of Opportunity in Supervised Learning." In *Proceedings of the International Conference on Neural Information Processing Systems,* 3323–3331.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. "Deep residual learning for image recognition." In *Proceedings of the IEEE conference on computer vision and pattern recognition,* 770–778. https://doi.org/https://doi.org/10.48550/arXiv.1512.03385.

He, Warren, Bo Li, and Dawn Song. 2018. "Decision Boundary Analysis of Adversarial Examples." In *Proceedings of the International Conference on Learning Representations.*

Heidemann, Julia, Mathias Klier, and Florian Probst. 2010. "Identifying Key Users in Online Social Networks: A PageRank Based Approach." In *Proceedings of the International Conference on Information Systems,* 79. January.

Hendrycks, Dan, and Kevin Gimpel. 2018. *A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks.* arXiv: 1610.02136 [cs.NE].

Hendrycks, Dan, Norman Mu, Ekin D Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. 2019. "AugMix: A simple data processing method to improve robustness and uncertainty." *arXiv preprint arXiv:1912.02781,* https://doi.org/https://doi.org/10.48550/arXiv.1912.02781.

Hernández-García, Alex, and Peter König. 2018. "Data augmentation instead of explicit regularization." *arXiv preprint arXiv:1806.03852,* https://doi.org/https://doi.org/10.48550/arXiv.1806.03852.

Hinton, Geoffrey, Oriol Vinyals, and Jeffrey Dean. 2015. "Distilling the Knowledge in a Neural Network." In *NIPS Deep Learning and Representation Learning Workshop.* http://arxiv.org/abs/1503.02531.

Hohman, Fred, Haekyu Park, Caleb Robinson, and Duen Horng Chau. 2020. "Summit: Scaling Deep Learning Interpretability by Visualizing Activation and Attribution Summarizations." *IEEE Transactions on Visualization and Computer Graphics,* https://fredhohman.com/summit/.

How Valuable Is The First Page of Google? https://yourwebedge.com/valuable-first-page-google/. (Accessed on 02/28/2020).

Hsu, Chin-Chi, Yi-An Lai, Wen-Hao Chen, Ming-Han Feng, and Shou-De Lin. 2017. "Unsupervised Ranking Using Graph Structures and Node Attributes." In *Proceedings of the ACM International Conference on Web Search and Data Mining,* 771–779.

Huang, Ling, Anthony D. Joseph, Blaine Nelson, Benjamin I.P. Rubinstein, and J. D. Tygar. 2011. "Adversarial Machine Learning." In *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence,* 43–58. AISec '11. Chicago, Illinois, USA: Association for Computing Machinery. https://doi.org/10.1145/2046684.2046692.

Ilyas, Andrew, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. 2019. "Adversarial examples are not bugs, they are features." *Advances in neural information processing systems,* https://doi.org/https://doi.org/10.48550/arXiv.1905.02175.

Inoue, Hiroshi. 2018. "Data augmentation by pairing samples for images classification." *arXiv preprint arXiv:1801.02929,* https://doi.org/https://doi.org/10.48550/arXiv.1801.02929.

Ioffe, Sergey, and Christian Szegedy. 2015. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." In *International conference on machine learning,* 448–456. pmlr. https://doi.org/https://doi.org/10.48550/arXiv.1502.03167.

Jagielski, Matthew, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. 2018. "Manipulating machine learning: Poisoning attacks and countermeasures for regression learning." In *Proceedings of the IEEE Symposium on Security and Privacy,* 19–35. IEEE.

Jia, Jinyuan, Binghui Wang, Xiaoyu Cao, and Neil Zhenqiang Gong. 2020. *Certified Robustness of Community Detection against Adversarial Structural Perturbation via Randomized Smoothing.* arXiv: 2002.03421 [cs.CR].

Jung, Alexander B., Kentaro Wada, Jon Crall, Satoshi Tanaka, Jake Graving, Christoph Reinders, Sarthak Yadav, et al. 2020. *imgaug.* https://github.com/aleju/imgaug. Online; accessed 01-Feb-2020.

Kamishima, Toshihiro, Shotaro Akaho, and Hideki Asoh. 2012. "Enhancement of the neutrality in recommendation." In *Proceedings of the 2nd Workshop on Human Decision Making in Recommender Systems,* 8–14.

Kang, Guoliang, Xuanyi Dong, Liang Zheng, and Yi Yang. 2017. "Patchshuffle regularization." *arXiv preprint arXiv:1707.07103,* https://doi.org/https://doi.org/10.48550/arXiv.1707.07103.

Kang, Jian, Jingrui He, Ross Maciejewski, and Hanghang Tong. 2020. "InFoRM: Individual Fairness on Graph Mining." In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* 379–389.

Kang, Jian, and Hanghang Tong. 2019. "N2N: Network Derivative Mining." In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management,* 861–870. CIKM '19. Beijing, China: ACM. https://doi.org/10.1145/3357384.3357910.

Kang, Jian, Meijia Wang, Nan Cao, Yinglong Xia, Wei Fan, and Hanghang Tong. 2018. *AURORA: Auditing PageRank on Large Graphs.* arXiv: 1803.05068 [cs.SI].

Kilbertus, Niki, Mateo Rojas-Carulla, Giambattista Parascandolo, Moritz Hardt, Dominik Janzing, and Bernhard Schölkopf. 2017. "Avoiding Discrimination through Causal Reasoning." In *Proceedings of the International Conference on Neural Information Processing Systems,* 656–666.

Kleinberg, Jon, Sendhil Mullainathan, and Manish Raghavan. 2016. *Inherent Trade-Offs in the Fair Determination of Risk Scores.* arXiv:1609.05807. eprint: arXiv:1609.05807.

Kleinberg, Jon M. 1999. "Authoritative Sources in a Hyperlinked Environment." *J. ACM* (New York, NY, USA) 46, no. 5 (September): 604–632. https://doi.org/10.1145/324133.324140.

Kleindessner, Matthäus, Samira Samadi, Pranjal Awasthi, and Jamie Morgenstern. 2019. "Guarantees for Spectral Clustering with Fairness Constraints." In *Proceedings of the International Conference on Machine Learning,* 97:3458–3467.

Kornblith, Simon, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. 2019. "Similarity of neural network representations revisited." In *International Conference on Machine Learning,* 3519–3529. PMLR. https://doi.org/https://doi.org/10.48550/arXiv.1905.00414.

Krause, Josua, Aritra Dasgupta, Jordan Swartz, Yindalon Aphinyanaphongs, and Enrico Bertini. 2017. "A Workflow for Visual Diagnostics of Binary Classifiers using Instance-Level Explanations." In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology,* 162–172. eprint: 1705.01968.

Krause, Josua, Adam Perer, and Kenney Ng. 2016. "Interacting with predictions: Visual inspection of black-box machine learning models." In *Proceedings of the CHI Conference on Human Factors in Computing Systems,* 5686–5697. ACM.

Krizhevsky, Alex, Vinod Nair, and Geoffrey Hinton. n.d. "CIFAR-10 (Canadian Institute for Advanced Research)," http://www.cs.toronto.edu/~kriz/cifar.html.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton. 2017. "ImageNet Classification with Deep Convolutional Neural Networks." Edited by F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger. *Communications of the ACM* 60 (6): 84–90. https://doi.org/https://dl.acm.org/doi/10.1145/3065386.

Kumar, Srijan, William L Hamilton, Jure Leskovec, and Dan Jurafsky. 2018. *Community Interaction and Conflict on the Web.* https://doi.org/10.1145/3178876.3186141.

Kwon, O., T. Crnovrsanin, and K. Ma. 2018. "What Would a Graph Look Like in this Layout? A Machine Learning Approach to Large Graph Visualization." *IEEE Transactions on Visualization and Computer Graphics* 24 (1): 478–488.

Lakkaraju, Himabindu, Ece Kamar, Rich Caruana, and Eric Horvitz. 2016. *Identifying Unknown Unknowns in the Open World: Representations and Policies for Guided Exploration.* arXiv: 1610.09064 [cs.AI].

Lakshminarayanan, Balaji, Alexander Pritzel, and Charles Blundell. 2017. "Simple and scalable predictive uncertainty estimation using deep ensembles." *Advances in neural information processing systems* 30.

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. 2015. "Deep learning." *Nature,* no. 7553, 436. https://doi.org/https://doi.org/10.1038/nature14539.

LeCun, Yann, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. 1998. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE* 86 (11): 2278–2324.

Lee, Kimin, Honglak Lee, Kibok Lee, and Jinwoo Shin. 2017. "Training confidence-calibrated classifiers for detecting out-of-distribution samples." *arXiv preprint arXiv:1711.09325.*

Lee, Kimin, Kibok Lee, Honglak Lee, and Jinwoo Shin. 2018. "A simple unified framework for detecting out-of-distribution samples and adversarial attacks." *Advances in neural information processing systems* 31.

Leskovec, Jure, and Julian J Mcauley. 2012. "Learning to discover social circles in ego networks." In *Advances in neural information processing systems,* 539–547.

Liang, Shiyu, Yixuan Li, and R. Srikant. 2020. *Enhancing The Reliability of Out-of-distribution Image Detection in Neural Networks.* arXiv: 1706.02690 `[cs.LG]`.

Lim, Soon Hoe, N. Benjamin Erichson, Francisco Utrera, Winnie Xu, and Michael W. Mahoney. 2021. *Noisy Feature Mixup.* https://doi.org/https://doi.org/10.48550/arXiv.2110.02180. arXiv: 2110.02180 `[cs.LG]`.

Liu, Mengchen, Shixia Liu, Hang Su, Kelei Cao, and Jun Zhu. 2018. "Analyzing the Noise Robustness of Deep Neural Networks." In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology.*

Liu, Mengchen, Jiaxin Shi, Zhen Li, Chongxuan Li, Jun Zhu, and Shixia Liu. 2016. "Towards better analysis of deep convolutional neural networks." *IEEE transactions on visualization and computer graphics* 23 (1): 91–100.

Liu, Shixia, Xiting Wang, Mengchen Liu, and Jun Zhu. 2017. "Towards better analysis of machine learning models: A visual analytics perspective." *Visual Informatics* 1 (1): 48–56. http://www.sciencedirect.com/science/article/pii/S2468502X17300086.

Liu, Yingqi, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. 2018. "Trojaning Attack on Neural Networks." In *Proceedings of the*

25th Annual Network and Distributed System Security Symposium. The Internet Society.

Liu, Yinhan, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. "RoBERTa: A Robustly Optimized BERT Pretraining Approach." *arXiv preprint arXiv:1907.11692,* https://doi.org/https://doi.org/10.48550/arXiv.1907.11692.

Lu, Junhua, Wei Chen, Yuxin Ma, Junming Ke, Zongzhuang Li, Fan Zhang, and Ross Maciejewski. 2017. "Recent progress and trends in predictive visual analytics." *Frontiers of Computer Science* 11, no. 2 (April): 192–207. https://doi.org/10.1007/s11704-016-6028-y.

Lu, Yafeng, Rolando Garcia, Brett Hansen, Michael Gleicher, and Ross Maciejewski. 2017. "The State-of-the-Art in Predictive Visual Analytics." *Computer Graphics Forum* 36 (3): 539–562.

Ma, Yuxin, Tiankai Xie, Jundong Li, and Ross Maciejewski. 2020. "Explaining Vulnerabilities to Adversarial Machine Learning through Visual Analytics." *IEEE Transactions on Visualization and Computer Graphics* 26 (1): 1075–1085. https://doi.org/10.1109/TVCG.2019.2934631.

Maaten, Laurens van der, and Geoffrey Hinton. 2008. "Visualizing data using t-SNE." *Journal of Machine Learning Research* 9 (Nov): 2579–2605.

Madry, Aleksander, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. "Towards deep learning models resistant to adversarial attacks." *arXiv preprint arXiv:1706.06083,* https://doi.org/https://doi.org/10.48550/arXiv.1706.06083.

Malinin, Andrey, and Mark Gales. 2018. "Predictive uncertainty estimation via prior networks." *Advances in neural information processing systems* 31.

Martin, Charles H, Tongsu Peng, and Michael W Mahoney. 2021. "Predicting trends in the quality of state-of-the-art neural networks without access to training or testing data." *Nature Communications,* no. 1, 4122. https://doi.org/https://doi.org/10.1038/s41467-021-24025-8.

Martinez, Clara Marina, Mira Heucke, Fei-Yue Wang, Bo Gao, and Dongpu Cao. 2018. "Driving style recognition for intelligent vehicle control and advanced driver assistance: A survey." *IEEE Transactions on Intelligent Transportation Systems* 19 (3): 666–676.

Mehrabi, Ninareh, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2019. *A Survey on Bias and Fairness in Machine Learning.* arXiv: 1908.09635 [cs.LG].

Mei, Shike, and Xiaojin Zhu. 2015. "Using Machine Teaching to Identify Optimal Training-set Attacks on Machine Learners." In *Proceedings of the 29th AAAI Conference on Artificial Intelligence,* 2871–2877.

Molnar, Christoph. 2020. *Interpretable machine learning.* Lulu. com.

Mozaffari-Kermani, M., S. Sur-Kolay, A. Raghunathan, and N. K. Jha. 2015. "Systematic Poisoning Attacks on and Defenses for Machine Learning in Healthcare." *IEEE Journal of Biomedical and Health Informatics* 19, no. 6 (November): 1893–1905. https://doi.org/10.1109/JBHI.2014.2344095.

Mu, Norman, and Justin Gilmer. 2019. "MNIST-C: A Robustness Benchmark for Computer Vision." *ArXiv,* https://doi.org/https://doi.org/10.48550/arXiv.1906.02337.

Netzer, Yuval, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. 2011. "Reading digits in natural images with unsupervised feature learning."

Ng, Andrew Y, Alice X Zheng, and Michael I Jordan. 2001. "Link analysis, eigenvectors and stability." In *International Joint Conference on Artificial Intelligence,* 17:903–910. Lawrence Erlbaum Associates Ltd.

Page, Lawrence, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank Citation Ranking: Bringing Order to the Web.* Technical Report 1999-66. Stanford InfoLab, November.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, et al. 2011. "Scikit-learn: Machine Learning in Python." *Journal of Machine Learning Research* 12:2825–2830.

Pedreschi, Dino, Salvatore Ruggieri, and Franco Turini. 2009. "Measuring discrimination in socially-sensitive decision records." In *Proceedings of the SIAM International Conference on Data Mining,* 581–592.

Pedreshi, Dino, Salvatore Ruggieri, and Franco Turini. 2008. "Discrimination-aware data mining." In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* 560–568.

Pitoura, Evaggelia, Panayiotis Tsaparas, Giorgos Flouris, Irini Fundulaki, Panagiotis Papadakos, Serge Abiteboul, and Gerhard Weikum. 2018. "On Measuring Bias in Online

Information." *Special Interest Group on Management of Data Record* 46, no. 4 (February): 16–21.

Radford, Alec, Luke Metz, and Soumith Chintala. 2015. "Unsupervised representation learning with deep convolutional generative adversarial networks." *arXiv preprint arXiv:1511.06434,* https://doi.org/https://doi.org/10.48550/arXiv.1511.06434.

Rahman, Tahleen, Bartlomiej Surma, Michael Backes, and Yang Zhang. 2019. "Fairwalk: Towards fair graph embedding." In *Proceedings of the International Joint Conferences on Artifical Intelligence,* 3289–3295.

Rahnama, Arash, Andre T Nguyen, and Edward Raff. 2020. "Robust design of deep neural networks against adversarial attacks based on lyapunov theory." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition,* 8178–8187.

Ren, Donghao, Saleema Amershi, Bongshin Lee, Jina Suh, and Jason D Williams. 2017. "Squares: Supporting interactive performance analysis for multiclass classifiers." *IEEE Transactions on Visualization and Computer Graphics* 23 (1): 61–70. https://doi.org/10.1109/TVCG.2016.2598828.

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. 2015. *U-Net: Convolutional Networks for Biomedical Image Segmentation.* arXiv: 1505.04597 [cs.CV].

Shafahi, Ali, W. Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. 2018. "Poison Frogs! Targeted Clean-label Poisoning Attacks on Neural Networks." In *Proceedings of the 32nd International Conference on Neural Information Processing Systems,* 6106–6116.

Shalev, Gabi, Yossi Adi, and Joseph Keshet. 2019. *Out-of-Distribution Detection using Multiple Semantic Label Representations.* arXiv: 1808.06664 [stat.ML].

Shamsabadi, Ali Shahin, Ricardo Sanchez-Matilla, and Andrea Cavallaro. 2020. "ColorFool: Semantic Adversarial Colorization." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* June.

Shneiderman, B. 1996. "The eyes have it: a task by data type taxonomy for information visualizations." In *Proceedings of IEEE Symposium on Visual Languages,* 336–343. September. https://doi.org/10.1109/VL.1996.545307.

Shorten, Connor, and Taghi M Khoshgoftaar. 2019. "A survey on image data augmentation for deep learning." *Journal of big data,* no. 60, 1–48. https://doi.org/https://doi.org/10.1186/s40537-019-0197-0.

Simonyan, Karen, and Andrew Zisserman. 2014. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556,* https://doi.org/https://doi.org/10.48550/arXiv.1409.1556.

Singh, Ashudeep, and Thorsten Joachims. 2018. "Fairness of Exposure in Rankings." In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining,* 2219–2228. KDD '18. London, United Kingdom: Association for Computing Machinery. https://doi.org/10.1145/3219819.3220088.

Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. "Dropout: a simple way to prevent neural networks from overfitting." Edited by Yoshua Bengio. *Journal of Machine Learning Research,* no. 56 (June): 1929–1958. https://doi.org/https://dl.acm.org/doi/10.5555/2627435.2670313.

Steinhardt, Jacob, Pang Wei Koh, and Percy Liang. 2017. "Certified Defenses for Data Poisoning Attacks." In *Proceedings of the 31st International Conference on Neural Information Processing Systems,* 3520–3532.

Suciu, Octavian, Radu Marginean, Yigitcan Kaya, Hal Daume III, and Tudor Dumitras. 2018. "When Does Machine Learning FAIL? Generalized Transferability for Evasion and Poisoning Attacks." In *Proceedings of the USENIX Security Symposium,* 1299–1316.

Sühr, Tom, Sophie Hilgard, and Himabindu Lakkaraju. 2020. *Does Fair Ranking Improve Minority Outcomes? Understanding the Interplay of Human and Algorithmic Biases in Online Hiring.* arXiv:2012.00423. eprint: arXiv:2012.00423.

Sundararajan, Kalaivani, and Damon L Woodard. 2018. "Deep learning for biometrics: A survey." *ACM Computing Surveys* 51 (3): 65.

Thomas, Sam, and Nasseh Tabrizi. 2018. "Adversarial Machine Learning: A Literature Review." In *Proceedings of the Machine Learning and Data Mining in Pattern Recognition,* edited by Petra Perner, 324–334. Springer International Publishing.

Tsioutsiouliklis, Sotiris, Evaggelia Pitoura, Panayiotis Tsaparas, Ilias Kleftakis, and Nikos Mamoulis. 2020. *Fairness-Aware PageRank.* arXiv:2005.14431. eprint: arXiv : 2005 . 14431.

Tzeng, F-Y, and K-L Ma. 2005. *Opening the black box-data driven visualization of neural networks.* IEEE.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. "Attention is all you need." *Ad-*

vances in neural information processing systems, https://doi.org/https://doi.org/10.48550/arXiv.1706.03762.

Vorobeychik, Yevgeniy, and Murat Kantarcioglu. 2018a. *Adversarial Machine Learning.* Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.

———. 2018b. "Adversarial machine learning." *Synthesis Lectures on Artificial Intelligence and Machine Learning* 12 (3): 1–169.

Wang, Alex, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. "GLUE: A multi-task benchmark and analysis platform for natural language understanding." *arXiv preprint arXiv:1804.07461,* https://doi.org/https://doi.org/10.48550/arXiv.1804.07461.

Wang, Q., Z. Xu, Z. Chen, Y. Wang, S. Liu, and H. Qu. 2021. "Visual Analysis of Discrimination in Machine Learning." *IEEE Transactions on Visualization and Computer Graphics* 27 (2): 1470–1480.

Wang, Xiaoyun, Minyhao Cheng, Joe Eaton, Cho-Jui Hsieh, and Felix Wu. 2018. *Attack Graph Convolutional Networks by Adding Fake Nodes.* arXiv: 1810.10751 [cs.LG].

We Analyzed 5 Million Google Search Results. Here's What We Learned About Organic CTR. https://backlinko.com/google-ctr-stats. (Accessed on 02/28/2020).

Wexler, James, Mahima Pushkarna, Tolga Bolukbasi, Martin Wattenberg, Fernanda Viegas, and Jimbo Wilson. 2019. "The What-If Tool: Interactive Probing of Machine Learning Models." *IEEE Transactions on Visualization and Computer Graphics* 26, no. 1 (August): 56–65. https://doi.org/10.1109/TVCG.2019.2934619.

Xiao, Chang, and Changxi Zheng. 2019. "One Man's Trash is Another Man's Treasure: Resisting Adversarial Examples by Adversarial Examples." *CoRR* abs/1911.11219. arXiv: 1911.11219. http://arxiv.org/abs/1911.11219.

Xiao, Han, Huang Xiao, and Claudia Eckert. 2012. "Adversarial Label Flips Attack on Support Vector Machines." In *Proceedings of the European Conference on Artificial Intelligence,* 870–875. Montpellier, France. https://doi.org/10.3233/978-1-61499-098-7-870.

Xie, T., Y. Ma, H. Tong, M. T. Thai, and R. Maciejewski. 2021. "Auditing the Sensitivity of Graph-based Ranking with Visual Analytics." *IEEE Transactions on Visualization and Computer Graphics* 27 (2): 1459–1469.

Xie, Tiankai, Yuxin Ma, Jian Kang, Hanghang Tong, and Ross Maciejewski. 2021. "FairRankVis: A Visual Analytics Framework for Exploring Algorithmic Fairness in Graph Mining Models." *IEEE Transactions on Visualization and Computer Graphics* 28 (1): 368–377.

Xie, Tiankai, Yuxin Ma, Hanghang Tong, My T. Thai, and Ross Maciejewski. 2021. "Auditing the Sensitivity of Graph-based Ranking with Visual Analytics." *IEEE Transactions on Visualization and Computer Graphics* 27 (2): 1459–1469. https://doi.org/10.1109/TVCG.2020.3028958.

Xu, Ke, Shunan Guo, Nan Cao, David Gotz, Aiwen Xu, Huamin Qu, Zhenjie Yao, and Yixin Chen. 2018. "ECGLens: Interactive Visual Exploration of Large Scale ECG Data for Arrhythmia Detection." In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems.* CHI '18. Montreal QC, Canada: Association for Computing Machinery. https://doi.org/10.1145/3173574.3174237.

Yang, Ke, and Julia Stoyanovich. 2017. "Measuring Fairness in Ranked Outputs." In *Proceedings of the International Conference on Scientific and Statistical Database Management.*

Yang, Yaoqing, Liam Hodgkinson, Ryan Theisen, Joe Zou, Joseph E Gonzalez, Kannan Ramchandran, and Michael W Mahoney. 2021. "Taxonomizing local versus global structure in neural network loss landscapes." *Advances in Neural Information Processing Systems,* 18722–18733. https://doi.org/https://doi.org/10.48550/arXiv.2107.11228.

Yang, Yaoqing, Rajiv Khanna, Yaodong Yu, Amir Gholami, Kurt Keutzer, Joseph E Gonzalez, Kannan Ramchandran, and Michael W Mahoney. 2020. "Boundary thickness and robustness in learning models." *Advances in Neural Information Processing Systems,* 6223–6234. https://doi.org/https://doi.org/10.48550/arXiv.2007.05086.

Yao, Sirui, and Bert Huang. 2017. "Beyond Parity: Fairness Objectives for Collaborative Filtering." In *Proceedings of the International Conference on Neural Information Processing Systems,* 2925–2934.

Yao, Zhewei, Amir Gholami, Kurt Keutzer, and Michael W Mahoney. 2020. "PyHessian: Neural networks through the lens of the hessian." In *2020 IEEE international conference on big data (Big data),* 581–590. IEEE. https://doi.org/https://doi.org/10.48550/arXiv.1912.07145.

Yuan, Jun, Changjian Chen, Weikai Yang, Mengchen Liu, Jiazhi Xia, and Shixia Liu. 2021. "A survey of visual analytics techniques for machine learning." *Computational Visual Media* 7 (1): 3–36. https://doi.org/https://doi.org/10.1007/s41095-020-0191-7.

Zehlike, Meike, Francesco Bonchi, Carlos Castillo, Sara Hajian, Mohamed Megahed, and Ricardo Baeza-Yates. 2017. "FA*IR: A Fair Top-k Ranking Algorithm." In *Proceedings of the ACM on Conference on Information and Knowledge Management,* 1569–1578. https://doi.org/10.1145/3132847.3132938.

Zhang, Hongyang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. 2019. "Theoretically principled trade-off between robustness and accuracy." In *International conference on machine learning,* 7472–7482. PMLR. https://doi.org/https://doi.org/10.48550/arXiv.1901.08573.

Zhang, Hongyi, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. 2017. "mixup: Beyond empirical risk minimization." *arXiv preprint arXiv:1710.09412,* https://doi.org/https://doi.org/10.48550/arXiv.1710.09412.

Zhang, Jiawei, Yang Wang, Piero Molino, Lezhi Li, and David S. Ebert. 2019. "Manifold: A Model-Agnostic Framework for Interpretation and Diagnosis of Machine Learning Models." *IEEE Transactions on Visualization and Computer Graphics* 25 (1): 364–373. https://doi.org/10.1109/TVCG.2018.2864499. arXiv: 1808.00196.

Zhang, Quanshi, Ying Nian Wu, and Song-Chun Zhu. 2018. "Interpretable Convolutional Neural Networks." In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* June.

Zhao, Zhengyu, Zhuoran Liu, and Martha Larson. 2020. "Towards Large Yet Imperceptible Adversarial Image Perturbations With Perceptual Color Distance." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* June.

Zhong, Zhun, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. 2020. "Random erasing data augmentation." In *Proceedings of the AAAI conference on artificial intelligence,* 13001–13008. https://doi.org/https://doi.org/10.48550/arXiv.1708.04896.