Abstract—Machine learning (ML) algorithms have greatly improved the performances of many computationally expensive operations, including detection of frauds and attacks for security applications. However, these algorithms are often hard for users to interpret or modify due to their hidden processes. This paper presents an explainable visualization approach to provide a set of developer tools that enable users to apply ML algorithms efficiently without requiring users to understand the algorithms completely. We use an algorithm of convolutional neural network (CNN) for Wikipedia vandal detection as an example. Our highly coordinated visualizations serve as a visual analytics interface for developers to analyze the behaviors of algorithms and large-scale data. It enables users to study hidden relationships among the spaces of data, algorithm parameters, and results that are essential to understand the CNN detection mechanism. We provide several case studies to demonstrate how our approach can be used to explore the clusters of parameter space, identify outlier of parameters or data, study the parameter sensitivity ranges, and select suitable ranges of parameter for combined criteria (e.g., covering selected important cases and maintaining relative high success ratio) that are hard to describe with single objective functions. Due to the complexity of the problem, we also discuss the limitations of our approach and future work.

Index Terms—Explainable visualization, CNN, vandal detection, Wikipedia user behavior

I. INTRODUCTION

The advances of machine learning (ML) techniques have significantly improved the performance of many algorithms. For security applications, ML approaches have pushed the success ratios of vandal detection to numbers that have never been achieved before. However, they are often called a black box, as it is hard for people to understand the hidden procedure, which hinders users from understanding the algorithms, predicting the results, and adjusting the parameters to reach different combinations of criteria. Explainable visualization can fill in this gap to provide the right visual interfaces for developers to better interact with complex ML algorithms [12], [19].

Previous work has tried to open the black box by visualizing the changes of data at neuron levels [2], [13], [17], [18], [24], [33]. They are valuable for revealing the inside procedure of ML algorithms, however they often require prior knowledge of the ML models and modification of the project’s source code which is not necessarily available. Many ML projects built upon popular platforms, such as Tensorflow or Caffe, can not provide the detailed information requested by previous approaches. Therefore, this work focuses on providing new visual analytics tools for developing interactive exploration functions to suit for different needs of vandal detection tasks.

In this work, we present an approach to reveal the hidden relationships between the input (data and parameters of ML algorithms) and the output (vandal detection results), with the goal of improving the understanding of what ML algorithms can achieve and how to adjust the results for desired objectives. Our approach embeds domain knowledge of users into the detection process through user interaction, which enriches the evaluation metrics of ML algorithms (currently success ratio is often used). For example, we may identify a group of vandal users that should be included in the detection results, even under the cost of lowering the overall success ratio. Another example is that we can study the types of users that are identified wrongly by the algorithm, explore their attributes, and further improve the detection algorithms.

We take the problem of Wikipedia vandal detection using convolution neural network (CNN) as an example. We have identified three spaces that matter to the process, including the parameter space, result space, and data attribute space. We have adopted a brute force method to build the connection among the spaces. Our system supports highly coordinated visualizations for interactive exploration of the hidden relationships. Specifically, we have designed slide window plot for parameter space, Sankey diagram for result space, and a two-layer scatter plot with hex-bin statistical view for visualizing large-scale data distributions for both parameter and data attribute spaces. Interactive exploration functions with several types of data flows are provided to support users to analyze the problem by adjusting any criteria from the interface.

Our approach serves as an outer wrapper of ML algorithms that are involved of data, input and output. It is an interface to visualize the connections among previously separated spaces, as well as a tool for inputting combined criteria that are generally missing from ML algorithms. We provide several case studies to demonstrate how our approach can be used to improve the understanding of vandal detection algorithm, including exploring the clusters of parameter and data spaces, identifying outlier among them, studying the parameter sensitivity ranges, and providing an interactive tool for selecting combined criteria that are hard to describe with
The remainder of this paper is organized as follows. We first present the related work on explainable visualization, visualization of parameter space and vandal detection in Section II. Section III introduces the example CNN algorithm for Wikipedia vandal detection. Section IV describes our approach of explainable visualization and Section V presents the system of interactive exploration. Section VI provides several case studies to demonstrate example results and our discussions. Section VII concludes the paper and describes the future work.

II. RELATED WORK

We describe the related work from the aspects of explainable visualization for ML algorithms, visualization of parameter space and Wikipedia vandal detection respectively.

A. Explainable Visualization for ML Algorithms

With the fast development of ML techniques, a number of visualization work has been actively involved with ML in two ways: using visualization to improve understanding of ML algorithms and using ML to improve visualization.

We focus on the first type of work, using visualization to improve understanding of ML, as it is more related to out work. Liu et al. [19] classified the relevant work into three categories: understanding, diagnosis, and refinement. A recent survey summarized the recent literature on interactive ML and classified them into a task-oriented taxonomy [12]. For example, Talbot et al. [31] presented EnsembleMatrix which visualized the confusion matrices to help users understand relative merits of various classifiers. The evaluation results showed that users were able to quickly combine multiple classifiers operating on multiple feature sets to produce an ensemble classifier with high accuracy. Yosinski et al. [35] presented two tools of visualization for understanding deep learning: one visualized the activations produced on each layer of a trained convnet and the other visualized features at each layer of a DNN via regularized optimization in image space. Both are found useful for building valuable intuitions about how convnets work. From the visualization field, Choo and Liu [8] reviewed visual analytics, information visualization, and machine learning perspectives relevant to the topic of explainable visualization and discussed potential challenges and future research directions. Recently, a number of visualization techniques have been developed for CNN [2], [18], deep neural networks [13], [24], deep generative models [17], and deep Q-networks [33].

The second type of work has also attracted attentions of visualization researchers. ML algorithms have been used in different types of visualization and rendering tasks. For example, Leban et al. [16] presented VizRank to assess possible data projections and rank them by their ability to discriminate between classes visually. Kwon et al. [15] presented a ML approach of large graph visualization based on computing the topological similarity of graphs using graph kernels.

B. Visualization of Parameter Space

Visualization approaches for parameter spaces have also been explored, with similar goals of this work on gaining understanding of various algorithms. These techniques have been designed for a number of applications that often require expert inputs to control complex procedures, such as challenging problems of image analysis. For example, Schmucker [26] presented several basic visualization methods to visualize neuron parameters. Tam et al. [32] transformed multiple feature-based time-series for each expression in measurement space to a multi-dimensional representation in parameter space and visualized the correlation between the algorithm space and the goal classifying facial dynamics.

The common challenge is to handle the sampling problem among the parameter space, which creates a large number to infinite combinations of parameters. Some techniques were built upon clear understanding of the algorithms involved in the analysis process. For example, Pretorius et al. [25] developed an interactive visualization technique that enables users to analyze the relationships between sampled input parameters and corresponding output for biomedical image analysis framework. They analyzed conventional parameter optimization process for image analysis and formulated user requirements. Luboschik et al. [20] provided a visualization of a coarsely sampled subspace of the parameter space and its corresponding simulation outcome. They allowed users to identify regions for further drill-down and refine subsampling on the parameter space. In addition, Sedlmair et al. [27] provided a conceptual framework for visual parameter space analysis problems with a data flow model, navigation strategies, and a characterization of six analysis tasks.

This work combines the previous work on interactive ML and visualization of parameter space to develop an explainable visualization approach as developer tools. We share a similar goal with the previous methods on improving the understanding of complex algorithms.

C. Wikipedia Vandal Detection and Visualization

A number of vandal detection methods have been developed for Wikipedia data [30]. For example, Kumar et al. [14] used three machine learning models for classification: Wikipedia Vandal Behavior (WVB), Wikipedia Transition Probability Matrix (WTPM), and the combined Vandal Early Warning System (VEWS) for early detection of vandal users. Their user behavior identification is based on statistical analysis performed on already know vandal and benign users. Mola-Velasco [22] extracted several vandalism indicating features from edits in a vandalism corpus and used a supervised learning algorithm for vandal detection. The best performing classifiers were LogitBoost and Random Forest.

There are plenty of visualization approaches designed for analyzing various social network data [6], including visualizations of page ranks [34] and behaviors of participants [3]. Also related is the topic of network security as an application of visualization [28]. For example, network host behavior is visualized through positional changes in a two dimensional
space using a force-directed graph layout algorithm [1]. Suh et al. [29] proposed revert graph where a user is denoted as a node and a revert relationship as a link. Mödritscher et al. [21] also investigated user roles in collaboration by detecting and visualizing structural patterns. Hecking and Hoppe [10] proposed a node-link approach for visualizing dynamically evolving edits collaboration based on the article content. Flock and Acosta [9] presented a system that combines various visualization techniques to facilitated the understanding of intra-article disagreement between editors. Chan et al. [5] presented a web-based visualization system for collaborative platform such as Wikipedia through discovering, integrating, and visualizing additional data drawn from multiple Wikipedia articles, thus reducing the cost of data integration. Holloway et al. [11] proposed a system to analyze and visualize the semantic coverage of Wikipedia and its authors. Pang and Biuk-Aghai [23] introduced map-like Wiki Visualization to highlight the semantics coverage of Wikipedia. Chevalier et al. [7] integrated five visual indicators to Wikipedia layout in order to highlight the maturity and quality assessment of an article for the readers. Borra et al. [4] investigated the analysis and visualization of controversies in Wikipedia articles.

With relatively different objective of these approaches, this work focuses on developing explainable visualization system for understanding CNN-based vandal detection algorithms.

III. WIKIPEDIA VANDAL DETECTION SYSTEM

This section introduces our approach of Wikipedia vandal detection. Our main objective is to use this example to illustrate the challenges from applying ML algorithms, instead of exploring optimal vandal detection solutions.

A. CNN Vandal Detection of Wikipedia

We use the UMWDiagram dataset which contains a host of information about 34M pages and 23M users during a year long duration [14]. The major information is which user who edited which page at what time, if the edit was reverted, and the ground truth of if a user is benign or vandal. Our goal is to identify vandal users who malicious edited wrong information based simply on the editing behaviors of users.

For each user, we can collect a number of attributes, such as the number of pages a user has edited, the frequency of edits, and the type of Wikipedia pages a user has edited. To illustrate the classification problem, we have selected the following features from [14], which are shown to be effective on distinguishing benign and vandal users. These feature attributes are used in both CNN classification algorithm and our explainable visualization system for studying the correlations related to user properties.

We generate a matrix of users vs. features as the input to the CNN model. Specifically, we have chosen the following features to describe each user:

- Frequency of continuous edits. We collect continuous edits of a user on the same page and measure the time interval between the two continuous edits. A user may have multiple continuous edits on the same or different pages. Based on the frequency, we can define quick and slow continuous edits.
- The number of good (non-reverted) edits. It is used to define users with positive impacts when larger than or equal to a pre-defined threshold. We can assign value 1 for users with positive impacts and 0 for others.
- The number of bad (reverted) edits. Similar to above, it is used to define users with negative impacts when larger than or equal to a pre-defined threshold. We can assign value 1 for users with negative impacts and 0 for others.

Figure 1 shows that the pipeline of our CNN model includes two convolution layers, two pooling layers, two dropout layers, and a full connected layer. For each convolution layer, we adopt the ‘relu’ function as our activate function.

We divide the dataset into two parts – 70% for training and 30% for evaluation. Our model outputs 4 indications to evaluate its performance: true positive (TP) for correct detection of benign users, true negative (TN) for correct detection of vandal users, false positive (FP) for wrong results of benign users, and false negative (FN) for wrong results of vandal users.

B. Parameter Space

Corresponding to the features of users, our parameter space is composed with three dimensions:

- **Edit Frequency** $t_f$ is used to define “fast” and “slow” continuous edits.
- **Threshold of positive contribution** $t_p$ is defined by the number of good (non-reverted) edits larger than or equal to a pre-defined threshold.
- **Threshold of negative impact** $t_n$ is defined by the number of good (non-reverted) edits larger than or equal to a pre-defined threshold.

As a general approach, we make all parameters consistent to be in the range of $[0, 100]$ by scaling each parameter between the possible minimum and maximum values. For example, the minimum of positive impact $p_i$ is 0 and the maximum is the largest good edits among all the training data. As shown in Figure 2, the choices of parameters can affect the algorithm results significantly. The figure on the top left corner of each example maps parameter samples into a 2D domain, which illustrates the distribution of parameters to the classification results. We provide the detailed design in Section IV and use this example to demonstrate that parameter space can be further explored for understanding algorithm behaviors.

![Fig. 1. The pipeline of our CNN vandal detection algorithm.](image-url)
IV. EXPLAINABLE VISUALIZATION

The main challenges of applying our CNN and general ML algorithms come from the hidden processes and relationships between inputs and outputs. This section provides an overview of our visualization and describes the design of each part.

A. Approach Overview

Since our main goal is to enable interactive exploration of the parameter space based on vandal detection results, we design our approach to visualize all three relevant spaces: input, output and data attribute spaces. To provide interactive analysis functions among these spaces, we first develop a brute-force mechanism covering a large number of combinations in the parameter space. For each parameter combination, we run the CNN model and collect classification results for each user and the overall detection performance.

As shown in Figure 3, our visualization interface contains three main components, which are described in details in the following. We have also developed a highly coordinated interaction system to study various hidden relationships among the spaces, which is described in Section V.

B. Visualization of Parameter Space

The parameter space is constructed by all the parameters, each representing a dimension. Depending on the type of parameters, the parameter space can be continuous or discrete. In our work, it is a continuous 3D space composed of \( t_f \), \( p_i \) and \( n_i \) dimensions. We uniformly sample the parameter space, such as generating 20 values from each parameter, which gives \( 20^3 = 8000 \) parameter combinations. We could reduce the sample size by introducing multi-scale techniques.

The number of parameter combinations increases exponentially with the number of parameters and the frequency of sampling. To provide both visualization and interaction functions at different granularity levels, we have designed two components of this view. As shown in Figure 3, the scatter plot on the top provides an overview of the entire parameter space and the slide window plot on the bottom shows detailed values for choosing individual value ranges.

The overview of parameter samples is generated with a two-layer approach, as shown in Figure 4. The top layer projects all samples onto a 2D domain, which is independent of the number of parameters and can be generally applied to different algorithms. Each point represents one combination of our parameters and corresponds to one result in the other two spaces. We project parameter samples by applying principle component analysis (PCA) on the matrix of parameter samples vs. classification results of users. We keep 90\% components for PCA and use the first 2 dimensions of the PCA to layout parameter samples. This approach can be suited for problems with arbitrary parameters (more than 1 parameter). The colors of nodes represent success ratios of classification—good(60%-80%), poor(0%-40%), average(40%-60%). For large dataset, we randomly pick a set of samples to reduce the rendering cost. Users can pick one or more certain combination of parameters by an interaction of lasso.

When there are a large number of nodes, the scatter plot may be overcrowded. Our background layer is generated to provide a high-level statistical information of node distribution. In our approach, we divide the whole space into hex-bins and count the number of nodes in each hex-bin from the top layer. The colors of hex-bin can be used to visualize different values, such as the density of nodes in each region or the major classification result.

While the overview panel visualizes the distribution of parameter samples, we also provide a slide window plot on the bottom. This plot provides a slide window for each parameter in the system. One purpose is to select the values or ranges of parameters interactively by moving or adjusting the sizes of a window for each parameter, triggering our system to update the other spaces correspondingly. When multiple parameter samples are selected from other visualization panels, we adjust each window plot to reflect the entire range of parameter values. We also add points to show the involved values of parameters, as shown in parameter space of Figure 3.

C. Visualization of Result Space

The result space is composed of several types of statistical results correspond to our CNN model, including TP, TN, FP, and FN. As shown in Figure 3, Sankey diagram is used to visualize our result space. The advantage of Sankey diagram over other visualization methods, such as parallel coordinates,
Fig. 3. Our system interface. There are three main components in our user interface, input parameter space (left), result space (middle), and the user/attribute space (right). Interactive exploration functions are provided to study various hidden relationships among them for improving vandal detection algorithms.

Fig. 4. Visualization and interaction of parameter space. We provide two ways for users to decide the range of each parameter.

is to visualize the distribution of results flexibly with the curved strokes. The Sankey diagram is composed of axis, segment set, link set and link weights. Specifically, each vertical axis represents one of the TP, TN, FP, FN results. We divide each axis into multiple intervals with the same size, such as 20 segments with step 5. Each interval is represented as a segment, such as TP_{20} indicating the TP value range of (15, 20]. The length of each segment represents the number of results that fall into its interval. We also calculate the number of parameter combinations belong to each link and adjust the link weights.

This component is to visualize the result of the CNN model and enable users to specify the ranges of each ideal result range. It also helps users to find suitable combinations of parameters by filtering information from other spaces. The interaction can be achieved by either clicking the diagram segments directly or choosing ‘good’, ‘average’, and ‘poor’ result groups shown as check boxes on the side.

D. Visualization of Attribute Space

Similar to the parameter space, the number of attributes can also vary from problems. To develop a general approach, we use the same design as the overview visualization of parameter space to visualize the distribution of attributes. As shown in Figure 5, our approach contains two layers which can be flexibly applied since users may have various features in different applications. The top layer visualizes the distributions of users based on their attributes. We also apply PCA to layout all the user nodes. The node colors represent the classification results in the four categories and the sizes of node radius are decided by their individual features which extracted from their feature matrix. The background layer provides a statistical view of attribute distribution inside each hex-bin region.

We also provide several interactions for selecting users based on attribute values. Firstly, users can choose the interested group with the lasso-interaction. Secondly, user groups can be filtered by the result categories or attribute ranges, represented as check boxes on the right side of the panel.

V. INTERACTIVE ANALYSIS SYSTEM

We have developed an interactive analysis system integrating the CNN detection algorithm and our explainable visualization approach. This section first presents several types of data flows that our system supports for interactive analysis, and then our system architecture and performance.

A. Interactive Analysis

To support flexible interactive analysis of the data, attribute, and parameter spaces, we provide interaction methods for each visualization as well as three interaction schemes that link all the spaces. The procedures of interactive analysis are often achieved by combining individual interaction functions until reaching for final conclusions. The following presents three important data flows our system provides.

Exploring Parameter Space. We support exploration of the parameter space by initializing interaction from either
Fig. 5. Our two-layer visualization for handling large-scale data with a background hex-bin layout for showing data statistics at local regions and foreground nodes for showing data samples. Example options: (a) background for data density and foreground for data nodes; (b) background for data density and foreground for two types of attributes; (c) background for majority category and foreground for two types of attributes. This method is used for both the top panel of parameter space and data attribute space.

Fig. 6. Three types of data flows to support interactive analysis tasks.

Fig. 7. Architecture of our system. The four core components are the back-end application, the database, the Redis for cache, and the front-end application.

Exploring Result Space. We allow users to choose any value range from each of the four categories of TP, TN, FP, and FN in the result space. Our system then finds all satisfying parameter combinations and show them in both panels of the parameter space. This interaction is useful for identify the good ranges of parameters for high successful ratios. Users can continue to explore individual parameter set and study their effects on the user attributes.

Exploring Attribute Space. The attribute space visualizes data according to the detection results or data attributes. We can select a group of users or an individual user with the provided interaction functions, such as nodes with abnormal sizes for different attribute values. Our system searches for the parameter combinations that lead to the best successful detection ratios for these selected users. We can continue to filter parameter sets and distinguish their results. This interaction is useful to study the suitable ranges of parameters and successful ratios for different types of users, therefore providing information to improve the detection algorithm.

B. Prototype System

As shown in Figure 7, apart from the front-end application in which we implement our visualization system, the back-end application plays very important role in our project. It is responsible to process data-set, train our CNN algorithm model, and provide application program interface (API) for our front-end application.

Data processing module. This module is to provide the training and testing data-set for our CNN algorithm model for each parameter set. For each parameter combination, we generate the corresponded feature matrix from the Wikipedia data and target matrix for our CNN model.
Deep learning module. We have implemented our CNN algorithm base on Tensorflow and Keras. During an one-time offline process, we pre-train all the data samples for different parameter sets and save results in a database.

Web module. Our system uses HTTP+JSON to exchange data between back-end application and front-end application. To support this requirement, we develop our web module base on Django, a popular web framework in Python Community. In this module, we expose several APIs to interact with our front-end application.

C. Quantitative Results

In our system, we collect log data and analyze performance of the core components. All the statistics we collect, shown in Table 1, are based on the running environment of CPU (Intel i7-6700K 4.00GHZ, 6 Core), RAM (16G), and OS (Windows Server 2012 64-bit).

In general, our system provides good performance for interactive exploration, as all the response times of interactive components are within 3 secs. From the results, we can see the most time-consuming task in our system is the offline component – training and collecting all CNN results. Since this is only performed once and the results are stored, we did not accelerate this step. The time costs for rendering parameter space and attribute space are similar because both of them need to render thousands of points. These steps are accelerated in our system with the visualization design of two layers (hex-bin for statistics and nodes for samples).

VI. CASE STUDIES AND DISCUSSIONS

With the goal of providing interactive exploration tools, we demonstrate how our explainable visualization approach can be used for tasks related to restricted optimization, partition, outlier and sensitivity measurements. We also discuss the advantages and limitations from the case studies.

A. Partition of Parameter Space

The overview panel of our parameter space visualizes the distribution of parameter combinations, which may show clustered organizations. As shown in Figure 8, we randomly select three clusters in our parameter space to observe their corresponding results. For each cluster, we can observe different detection results, such as a very low FP in the middle figure. In addition, more obvious differences can be found in attribute space, such as the region highlighted by the rectangles in Figure 8. This case study demonstrates that our parameter space provides useful visual information to explore the partition of parameter space intuitively.

B. Outliers of Parameter Space or Attribute Space

Outliers are interesting since they may lead to special cases or issues of the algorithm. Since the overview of our parameter space is generated by data similarities, the nodes outside each cluster centers could be outliers. As shown in Figure 9, we select two outliers randomly in the parameter space. The result space reveals very different behaviors – an average result (a) and a very good result (b).

Similarly, we can explore outliers in the attribute space. The selected yellow nodes in the attribute space can be successfully classified by the revealed parameter combinations. Since the values of the first two parameters are very different, this indicates that the optimal parameter sets supporting the correct prediction of such outliers are very different. In addition, it may suggest that these two user groups are hard to be detected successfully under one input condition.

C. Sensitivity Ranges of Parameters

Another aspect users concern most is which parameter is sensitive, which means that a small fluctuation may lead to very different results. As shown in Figure 10, we randomly select a range of parameters in the slide window plot in the parameter space, which gets a general result (TP:25, TN:40).

We demonstrate how we explore the sensitive parameter ranges interactively. As shown in Figure 10 (b), we keep the ‘good_param’ and ‘bad_param’ parameters while adjusting the ‘time_param’ in a small step, which leads to a significant change in the overall result (TP:40, TN:35). However, as shown in Figure 10 (c or d), when we keep the ‘time_param’ parameter instead of adjusting the ‘good_param’ or ‘bad_param’, the results are almost same as the (a). This may suggest that our algorithm is much more sensitive to the ‘time_param’ parameter.

We continue to explore the sensitive region for the ‘time_param’ parameter. We keep the range of the other two parameters and adjust ‘time_param’ in small steps. Finally we find several sensitive regions – [10,25],[40,55], and [70,90].

D. Restricted Optimization with User Interaction

For complex detection objectives that are hard to define, our approach provides an interactive interface to enforce additional restrictions. For example, we can specify a group of benign users to be classified as good users while maintaining the optimal detection TP+TN ratios. The restrictions can combine user interaction from all of our visualization spaces, thus achieving a variety of complex requirements.

In this case study, we show how users explore the optimal parameter sets base on their own demands. As shown in Figure 11 (a), we first use the check boxes of ‘true benign’ and ‘false benign’ in the attribute space to filter data. We then choose a user group with large number of edits shown with large node sizes. Our system automatically finds two recommended parameter sets that successfully classify this user group. We continue to find the optimal parameter combination by exploring them one by one. Figure 11 (b) shows the best
E. Discussion and Limitations

Depending on the available information, our approach can be viewed as an external wrapper to explain ML algorithms. Compared to internal approaches that can access information inside the ML architectures, such as at the neuron level, our approach only visualizes a subset of information. Since we focus on studying the relationships directly between the input parameters and output results, our approach is more effective for testing and applying ML algorithms.

While ML algorithms have significantly increases the performances of vandal detection, security applications often still rely on users to make the final decision, such as who are vandal users and should be removed from the system. The topic of explainable visualization is especially useful for security applications with unique sets of analysis tasks. Our approach does not post any special requirements on ML components and maintain ML as a block box; therefore it can be flexibly extended to similar vandal detection algorithms.

Our approach may be improved from several aspects. There are alternative designs of our visualization approach. The sampling of large parameter sets or large datasets can be improved. We may consider to provide additional layout algorithms to better reflect the cluster structures. The performance of the system can be further accelerated.

VII. CONCLUSION AND FUTURE WORK

This paper presents an explainable visualization approach to improve the understanding of ML algorithms, through bridging the connections among three important spaces of data and detection algorithm. We provide a visualization system with a set of rich interactive exploration functions for users to explore the differences of algorithm results under various combinations of input parameters and their effects on data with different attributes. Specifically, we use a CNN-based algorithm for detecting vandal users in Wikipedia as the example problem.
The case studies demonstrate that our system enables users to explore the sensitivity, outlier, and partition of algorithm results and provides new interactive ways for choosing optimal parameter combinations for combined detection results.

In the future, we would like to improve our system from several aspects. Firstly, we plan to improve the sampling strategy of our approach with dynamic approaches based on the interests of users. This can save a significant amount of computational resources and improve the sensitivities of parameter ranges. Secondly, we plan to provide additional interactive analysis functions in our visualization system. We plan to add a new component in our user interface to store the combinations of parameters corresponding to the link from the result space. Also, we plan to investigate another new component for users to explore various heterogeneous data spaces. Thirdly, we will explore other ML algorithms for vandal detection to improve the overall detection results. Last but not least, we plan to perform a formal user study to evaluate the effects of explainable visualization through a set of important vandal detection tasks. We will use the evaluation to improve our system.

REFERENCES


Fig. 11. Restricted optimization with user interaction. We interactively select a group of user from the attribute space (a) and explore the parameter combination that can successfully classify this group with the best overall detection results (b).


