VADAF: Visualization for Abnormal Client Detection and Analysis in Federated Learning

LINHAO MENG, YATING WEI, RUSHENG PAN, SHUYUE ZHOU, JIANWEI ZHANG, and WEI CHEN*, State Key Lab of CAD&CG, Zhejiang University, China

Federated Learning (FL) provides a powerful solution to distributed machine learning on a large corpus of decentralized data. It ensures privacy and security by performing computation on devices (which we refer to as clients) based on local data to improve the shared global model. However, the inaccessibility of the data and the invisibility of the computation make it challenging to interpret and analyze the training process, especially to distinguish potential client anomalies. Identifying these anomalies can help experts diagnose and improve FL models. For this reason, we propose a visual analytics system, VADAF, to depict the training dynamics and facilitate analyzing potential client anomalies. Specifically, we design a visualization scheme that supports massive training dynamics in the FL environment. Moreover, we introduce an anomaly detection method to detect potential client anomalies, which are further analyzed based on both the client model's visual and objective estimation. Three case studies have demonstrated the effectiveness of our system in understanding the FL training process and supporting abnormal client detection and analysis.

CCS Concepts: • **Human-centered computing → Visualization**; **Visual analytics**; **Visualization application domains**; • **Computing methodologies → Anomaly detection**.

Additional Key Words and Phrases: federated learning, visual analytics, anomaly detection

ACM Reference Format:
Linhao Meng, Yating Wei, Rusheng Pan, Shuyue Zhou, Jianwei Zhang, and Wei Chen. 2020. VADAF: Visualization for Abnormal Client Detection and Analysis in Federated Learning. *ACM Trans. Interact. Intell. Syst.* 1, 1, Article 1 (January 2020), 23 pages. https://doi.org/10.1145/3426866

1 INTRODUCTION

Federated Learning provides a powerful solution to decentralized machine learning training with privacy and security guarantee. Unlike the existing distributed machine learning algorithms designed for highly controlled environments, FL can be used among devices in an unbalanced and non-IID fashion. In a typical federated learning setting, training data remains locally on devices involved. The devices perform computation on their local data to improve a shared global model under a central server's coordination. The participating devices are generally numerous, and the training is an iterative process. These factors make it challenging to monitor complete training information. Besides, FL acts as a "black box" during the training process because of its non-transparent training data and invisible computation on devices. That is to say, the server in the FL setting has no access to the clients' training data, nor does it have complete control over the clients' behaviors.

*Corresponding author.

Authors' address: Linhao Meng, alice.menglh@gmail.com; Yating Wei, weiyating@zju.edu.cn; Rusheng Pan, panrusheng@zju.edu.cn; Shuyue Zhou, zhoushuyue@zju.edu.cn; Jianwei Zhang, zjw.cs@zju.edu.cn; Wei Chen, chenvis@zju.edu.cn, State Key Lab of CAD&CG, Zhejiang University, China, 866 Yuhangtang Rd, Hangzhou, 310058.

2160-6455/2020/1-ART1 $15.00
https://doi.org/10.1145/3426866

Malicious clients can easily attack the training process to influence the behavior of the global model. These clients which deviate from normal behaviors during the training process are considered as abnormal clients. However, the severity of abnormal clients in federated learning is not yet well-understood. Furthermore, the sophisticated working mechanism also puzzles ML experts in understanding federated learning and analyzing training failure or anomaly to further improve the model result.

Recently, researchers have studied federated optimization [28, 29], privacy enhancement [7, 40], and application scenarios [20]. However, there has been little studies in visualizing and analyzing the FL training process. Visualization is an effective tool to summarize complex data and recognize hidden patterns under big data. Visual analytics can be seen as an integral approach combining visualization, human knowledge, and data mining, facilitating the exploration and interpretation of complex data. The visualization of FL training dynamics can help ML experts monitor and understand the training progress. The visual analytics to abnormal clients is useful in discovering and analyzing the abnormal iteration or client. Compared with fully automatic abnormal client detection and treatment [34], the visual analytics system plays a part in understanding the training process and analyzing the abnormal behaviors with an advantageous combination of expert experience. One recent work [55] builds a demonstration system and tries to interpret how the FL system works in a demo case of a box car racing game. However, as an educational tool, the system shows limited information and a relatively simple visual design only suitable for a small number of clients. It still needs much consideration about how to show the abundant training dynamic and how to effectively utilize visual analytics to contribute to the detection and analysis of the abnormal clients in a typical FL setting.

Since federated learning is a novel distributed machine learning approach, the visual design and implementation for anomaly detection in FL pose unique challenges. Here we summarize them as follows:

C1. In a typical FL training scenario [19], a large number of devices participating in the FL training process iteratively produce numerous training metrics and model weights, making it difficult to present and analyze the training process from these massive data.

C2. Because of the inaccessibility of the training data and the invisibility of the computation, the FL training process on the client-side acts like a "black box". We cannot detect abnormal client and evaluate model from the data level referencing previous work [25, 50, 53].

C3. Various reasons may cause abnormal client behavior. It can be due to intentional malicious attackers disguised as normal clients [16] or unintentional client deficiencies like poor data quality [47, 60] and software/hardware defects. In addition, the non-IID property makes it possible that even the benign local model can be far from the current global model before the model converges, which aggravates the difficulty of identifying the attackers. Thus it is worth exploring different cases of abnormal clients for model improvement.

To tackle these challenges, we develop an interactive visual analysis system, VADAF, which visualizes the entire FL training process and empowers experts to drill down the abnormal clients spotted by means of the anomaly detection technique. Analyzing the training process is fulfilled by illustrating the dynamic training information at different levels of granularities. We design several interactive views to show training information from overview to detail and also from both perspectives of server and client (**C1**). Given the inaccessibility of training data, our system focuses on model-level representation and analysis. Our anomaly detection method uses trainable model parameters combined with statistical results of training metrics to detect potential abnormal clients (**C2**). To explore different cases of client anomaly, we present an approach to analyzing client anomaly using objective model evaluation results and visual model difference representation results with the whole interaction of our system(**C3**). In this paper, we consider two kinds of attacks against

federated learning, data poison, and model poison, and compare their effects with those of benign clients. We prove the effectiveness of our system by conducting experiments on MNIST [10, 32] and CIFAR-10 [30] using different model options. Our approach applies to classification tasks based on the horizontal federated learning architecture with no encryption.

To our best knowledge, this paper serves as a first attempt which combines visual analytics with anomaly detection to assist ML experts in observing the federated learning training process and analyze the abnormal clients. In summary, the main contributions of this paper are:

- A suite of visualization techniques for showing the rich dynamics of the FL training process.
- An anomaly detection scheme for numerous clients in a federated learning setting merely based on the model itself and training metrics instead of training data like previous works.
- A visual reasoning technique for potential client anomaly in FL, which can help explore different cases of client issues and measure the severity of abnormal clients.

2 BACKGROUND

In this section, we briefly introduce some basic concepts of federated learning and recent study about attacks against federated learning, which will be useful for subsequent discussions.

2.1 Federated Learning

Federated Learning is a decentralized machine learning approach proposed by Google [6, 19, 39]. It is proposed to solve the issue of data isolation and privacy in the era of big data. In reality, due to industry competition, data security, and other reasons, data is scattered in different enterprises or departments, existing in the form of isolated islands. Considering that artificial intelligence is data-driven, it is difficult to establish an effective model without a large amount of high-quality data. However, the traditional procedure of data integration has the risk of violating regulatory requirements and leaking users' privacy. In this case, how to design a machine learning framework that can make full use of the data of different parties under the conditions of data privacy and security is a major issue in the current artificial intelligence field.

Federated learning is proposed in the background mentioned above. The basic idea is to collaboratively construct a shared global machine learning model while training data that may contain private information remains distributed on devices like mobile phones. Yang et al. [57] categorize federated learning into three categories based on the distribution characteristics of the data. Our work focuses on the most commonly used horizontal federated learning in which devices share the same feature space but little overlap in the sample space. In a typical horizontal federated learning setting, there is one server and large numbers of participating devices. The training process is iterative, as shown in Fig. 1. At the beginning of each iteration, a random fraction of clients is selected. These selected clients download the current global model from the central server, perform local computation based on the current global model and local data, and then send the updates to the server. The server aggregates these locally-computed models with federated averaging algorithm [39] which is widely-used for federated model training to improve the shared global model:

$$w_t = \sum_{k=1}^{K} \frac{n_k}{n} w_t^k \tag{1}$$

where w_t is the shared global model parameters obtained by aggregating updated model parameters of the participating clients in tth iteration, K is the number of selected clients, n is the total amount of training data, n_k is the training data amount of client k, w_t^k is the model parameters after training on client k in tth iteration. Notice that except the averaged model parameters, the training metrics are also averaged and recorded, which help people know about the overall training progress. Then

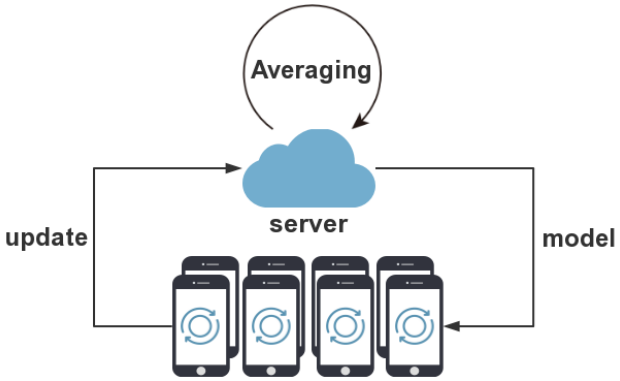

Fig. 1. Illustration of a federated learning architecture.

the next iteration begins, where w_t will be the initial model for each participating client. This training process continues till it reaches the predefined maximum number of training rounds or the model converges.

There are several advantages of this approach [39], which differentiate it from other distributed machine learning methods: 1) It is robust to unbalanced and non-IID data distributions. 2) It can significantly reduce the rounds of communication needed to train a good model in the decentralized setting. Some federated learning frameworks have been developed for experimentation with FL, such as TensorFlow Federated [17] and Federated AI Technology Enabler [54]. Meanwhile, there have been more and more FL architecture designs for different scenarios [57].

2.2 Attack Strategy

Federated learning guarantees data privacy by pushing model training on users' devices. It gives clients full control over local training but makes attackers easier to perform various attacks. Some attack strategies specifically against federated learning have been studied by scholars recently [14, 34]. Here we categorize the attack strategy into two types, data poison and model poison.

2.2.1 Data Poison. Since personal data is stored locally, a malicious client can attack the global model by poisoning the training data. There are usually two kinds of aims of the attack. One is to degrade the model's overall efficacy [2, 23]. This can be implemented by adding artificial fake data or shuffling the data label. We call this kind of attack *random poisoning attack*. The other one is to increase/decrease the probability of the model predicting a target instance as a target class [16]. Label-flipping can achieve this aim by flipping the labels of instances in one class to another class. Another approach aiming at the same goal is called backdoor [4, 12]. In this approach, some features or regions of the original training data are augmented and relabeled so that the model will be misled to classify the backdoor instances as a target label specified by the attacker. This kind of attack is called *targeted poisoning attack*.

2.2.2 Model Poison. The full control over the local training in federated learning gives the malicious client the possibility to manipulate the local model parameters directly and upload a malicious model to spoil the global model or achieve the backdoor attack [14, 52]. However, client selection and model aggregation mechanisms make small deviations of single client uneasy to cause significant damage to the global model. Drastically different model updates increase the risk of being detected by the server. Similar to data poison, model poison has its own strategies to make random poisoning attacks and targeted poisoning attacks. Sign-flipping achieves the attack by flipping the signs of

the local model updates [35, 56]. Additive noise approach adds Gaussian noise to the local model updates. Both attacks aim to make random poisoning attacks [35, 56]. There also has been some research about how to design a model replacement strategy to achieve targeted poisoning attacks [4].

3 RELATED WORK

3.1 Visual Analytics for Machine Learning

The monitor and analysis of the federated learning training process are essentially based on large volumes of logs for machine learning training on the clients. The general training dynamics representation, model evaluation approaches, etc., in visual analytics for machine learning are worth referring to in this work. In the training process of machine learning, there are generally large amounts of data sets involved and model parameters produced. It is a challenge itself to visualize these big volume data, let alone help experts analyze the training process and interpret the machine learning model with the help of visual interaction. Given the above facts, some work has been done on interactive visualization and visual analytics to understand and analyze machine learning models, which can assist experts in improving model performance and robustness. Most interactive machine learning tools achieve this aim by displaying multiple kinds of training information, including training data, features, metrics, and model performance, from which experts can get insight from the training process and debug the failure cases. For example, Ren et al. [48] design a performance visualization solution, Squares, for multiclass classification problems. Compared with the confusion matrix, it shows more details from the instance level, which helps practitioners analyze model performance faster and more accurately. But for general usage scenarios in which only the count-based metrics matter, Squares is a little complicated. Zhang et al. [59] propose a generic framework named Manifold to support debugging and comparison of different machine learning models in an interactive manner based on model performance and features extracted. Amershi et al. [3] present an interactive visualization system supporting performance analysis and debugging in machine learning. This system displays training information and graphs within a single compact visualization. Similarly, Kahng et al. [24] develop a visual analytics system, ACTIVIS, which is flexible in exploring many different complex deep neural network models for classification tasks. As deep learning grows, recent studies in visual analytics focus on some specific deep learning models, such as convolutional neural networks (CNNs) [36, 38], recurrent neural networks (RNNs) [42, 51] and deep generative models (DGMs) [37]. They try to visualize the model structure and introduce some manipulable visualization designs suitable for large volume data. Moreover, some of these work visualize and analyze models from the model level. For instance, Liu et al. [36] introduce some visual encodings to show the layer-level or even filter-level information for CNN models. They use a pixel chart to present filter-level information. The weakness is that it takes too much rendering space and resources, while the effective information is minimal. Besides that, TensorBoard [1], which provides a suite of visualization and tooling to track training dynamics, uses histograms to show model information like weights, biases, or other tensors. Rauber et al. [46] propose a point-based projection technique to explore the relationship between neurons and classes, and the similarity between neurons by means of dimensionality reduction. These works also prove the value of visualization feedback for model analysis and evaluation.

Federated Learning, as a new framework of training distributed machine learning models, attracts much attention of researchers in the field of machine learning. However, there has been little relevant visualization design to represent the FL training process. Wei et al. [55] propose a multi-agent visualization system to illustrate how a typical FL system works as an educational tool, but the design is relatively simple. The visual design of this system mainly includes several statistical charts,

such as histograms, radar charts, and line chart. These statistical charts are only suitable for a small number of clients. For the usual federal learning training scenarios with thousands of clients, the visualization design is not universal. The main difference between the existing visualization work of machine learning and the expected visualization of federated learning is that the latter requires consideration of the distributed environment in which a large number of clients involve in the training process and produce large amounts of data. In addition, training data is not available in the federated learning setting, which means we cannot use traditional methods to analyze the failure or anomaly in the federated learning training process from the data level. Nevertheless, the model-level design and analysis methods in the previous work can be used for reference. We also get inspiration about how to design visualization for general training logs and model evaluation results from the previous work.

3.2 Visualization for Distributed System

The FL training process is completed together with the participation of the clients and the server. Notice that Federated Learning is a distributed machine learning framework, the existing work about visualization for distributed systems has reference value for us in visual design. Gunter et al. [18] describe a system called NetLogger, which enables diagnosis and debugging in distributed systems. This system includes an interface for visualizing the log data and state, in which three types of graph primitives are introduced to represent different events in the distributed system. Similarly, Cosma et al. [13] present a visual solution for comprehending the design of distributed software systems, which uses colored rectangles to represent distributable feature cores. Beschastnikh et al. [5] design a new distributed system debugging tool, ShiViz, to visualize distributed-system executions. It combines time flow to display the distributed system's execution and different graphics to represent the events. These work focus on the display of the workflow in the distributed system, and in the meanwhile, they design multiple visual elements to represent different events or features. These visual element designs are helpful. However, in the FL setting, we do not care about different clients' workflow but only the training metrics and results sent to the server in each iteration. Kutzleb [31] creates a visual analytics tool to help domain experts explore data collected from distributed systems from the cluster level. All of the above work show the client information either from the single-client level or from the cluster level and lack an interactive tool to connect the views of different perspectives. In our work, we use small colored rectangles to represent participating clients and add different designs to indicate the different features or attributes of these clients. Furthermore, we apply an overview-to-detail visualization to support the display and exploration of all the clients' statistical data in each iteration and the specific client information.

3.3 Anomaly Detection

In the field of data analysis, it is often necessary to determine which instances stand out as being different from all others. Such instances are generally known as anomalies or outliers, and the goal of anomaly detection is to determine all such outliers. Anomaly detection can help us identify outliers during data preprocessing, and then we can remove the outliers or use some other methods to process these outliers, such as replacing the outliers with average values. Experimental results show that reducing the effect of the outliers in the data set can significantly improve the accuracy of the training results statistically.

When dealing with the client model sent to the server in each iteration during the FL training process, the model outliers raise concern, which may lead to bad aggregated model results. Therefore, it is necessary to take some measures to find these model outliers. Since the client models are not labeled as normal or abnormal, we need to use unsupervised anomaly detection methods to find the abnormal model. In an unsupervised anomaly detection algorithm, data is scored solely based

on its intrinsic properties, typically like distances [26, 27, 45] or densities [9], and then we can discriminate the anomalies according to the calculated scores and the set threshold. Inspired by this idea, we calculate the distances between client model vectors and aggregated model vector in each iteration for outlier detection by taking the server aggregated model as a baseline. Besides, statistical properties of data distribution such as mean, median, and quantiles can also be used for anomaly detection [22]. In this paper, we refer to the work of Leys [33], in which a method using median absolute deviation (MAD) is presented to detect outliers. We apply it in a half-normal distribution to find the outliers of the distance sequence. In addition to the above-mentioned statistical anomaly detection, there are many machine learning-based techniques for anomaly detection. For example, as an unsupervised learning method, clustering is an effective method to detect outliers, in which data instances that fall outside of these cluster groups could potentially be marked as anomalies [43]. However, generally speaking, the statistical analysis is applicable to various data sets for anomaly detection. Otherwise, it is much easier to interpret the anomalies from a statistical point of view. Similar work about malicious client detection in FL has been done by Li et al. [35], in which the central server applies spectral anomaly detection method to detect abnormal model updates and remove them in the aggregation process. This approach achieves good performance in improving model accuracy, but it requires additional calculations to train an encoder-decoder model using preprepared normal instances, which may not be representative of abundant training data.

There have been some relevant works about defense against client attacks in the FL setting [15]. Li et al. [34] use a pre-trained autoencoder model as the baseline to calculate the anomaly score of all the client models and use it to eliminate the anomalies' adverse impacts in the aggregation operation. Fung et al. [16] propose a defense method named FoolsGold, which adjusts the learning rate based on contribution similarity in the training process to defend sybil-based poisoning attacks. The above works focus on designing the defense strategy to mitigate the effect of malicious clients in the training process. However, the actual impact of malicious clients and the effect of these defense strategies in a non-IID setting are not well evaluated.

4 VADAF

In this section, we introduce our system, VADAF, from requirement analysis to system design. We also describe the data and model we use for training in this paper, as well as our anomaly detection method.

4.1 Requirement Analysis

VADAF was developed beginning with the identification of a few design requirements based on the authors' direct survey of federated learning and discussions with experts who have experience in FL. These high-level requirements also address the challenges proposed in the introduction section in a targeted manner. Here we summarize them as follows:

- **R1 - Providing an overview of the federated learning training process from both views of server and client.** As illustrated in **C1**, FL training is an iterative process in which a central coordinating server and a set of clients make a concerted effort to train a high-quality centralized model. This process produces large amounts of intermediate data, such as loss, accuracy, and the number of clients in one iteration, which can be used to specify the training progress and diagnose training failure. However, visualizing all the training dynamics will cause severe visual clutter. Considering the magnitude of intermediate data, we display the server and clients' overall statistics data as an overview of the FL training process. It can also serve as an entry point for analyzing specific iteration or client. This overview should also

help users spot certain iterations or clients of interest, which may show obvious anomaly on loss or accuracy.

- **R2 - Connecting the overall statistics with detailed training dynamics of numerous clients.** A high-level statistics overview is used as an entry point for the analysis of the FL process. We cannot get exhaustive information about some specific iteration or client from the statistics data. Therefore, as a complementary solution to **C1**, an overview-to-detail visualization is essential to connect the overview statistics with detailed training dynamics. By zooming in to the details, users can further examine iterations of interest and potential client issues from finer granularity.
- **R3 - Supporting detection of abnormal clients.** Anomaly detection is a technique aiming to find unusual patterns that do not conform to expected behaviors [11]. In the FL setting, clients train a model based on local data and send model updates to the server. The client training process acts like a "black box" because of the inaccessibility of training data and the invisibility of the computation, which also increases the risk of client attacks by sending malicious updates. For this reason, detecting client anomaly on the server-side is crucial for an aggregated optimal global model. This area is not thoroughly studied but worthwhile for FL research. Furthermore, as mentioned in **C2**, the inaccessible training data and invisible computation make it impossible to detect abnormal clients from the data level. We should do anomaly detection using training metrics and model parameters.
- **R4 - Supporting analysis of abnormal clients.** In the FL training process, distribution and non-IID property cause different clients to have totally different training progress. Clients that participate in less training or have some special but valued data may be treated as anomalies compared with other clients that have well-trained model results. Therefore, it is essential to analyze the abnormal clients and distinguish whether they are malicious or benign. Utilizing the analysis results, users can explore more simple and straightforward methods to distinguish different anomaly cases. This requirement is drawn by the challenge **C3**.

4.2 System Overview

The above-mentioned requirements motivate us to develop a visual analytics system, VADAF, for abnormal client detection and analysis in federated learning. This system includes three major modules:

- **A data storage module** that stores all the training logs of the server and clients in the federated learning training process.
- **An analysis module** that supports multiple statistical and analytical tasks, including abnormal client detection (**R3**) and analysis (**R4**).
- **An interactive visualization module** that discloses the training dynamics from different levels and perspectives and also provides users with a way of exploring and analyzing the training process (**R1, R2**).

As shown in Fig. 2, the pipeline of our system begins with the data storage module. We train our model in a simulated FL setting under the TensorFlow federated framework. The training logs are stored in MongoDB during the training process and organized into two parts. One is related to the client, which is produced directly in the client training process. It contains trainable model parameters(variables), performance metrics(loss and accuracy), and evaluation results. We evaluate each client model's performance using randomly selected data from the test dataset in each round and store the predictions and true labels to generate the confusion matrix for use in the visualization module. The other is acquired from the server-side and mainly includes two types

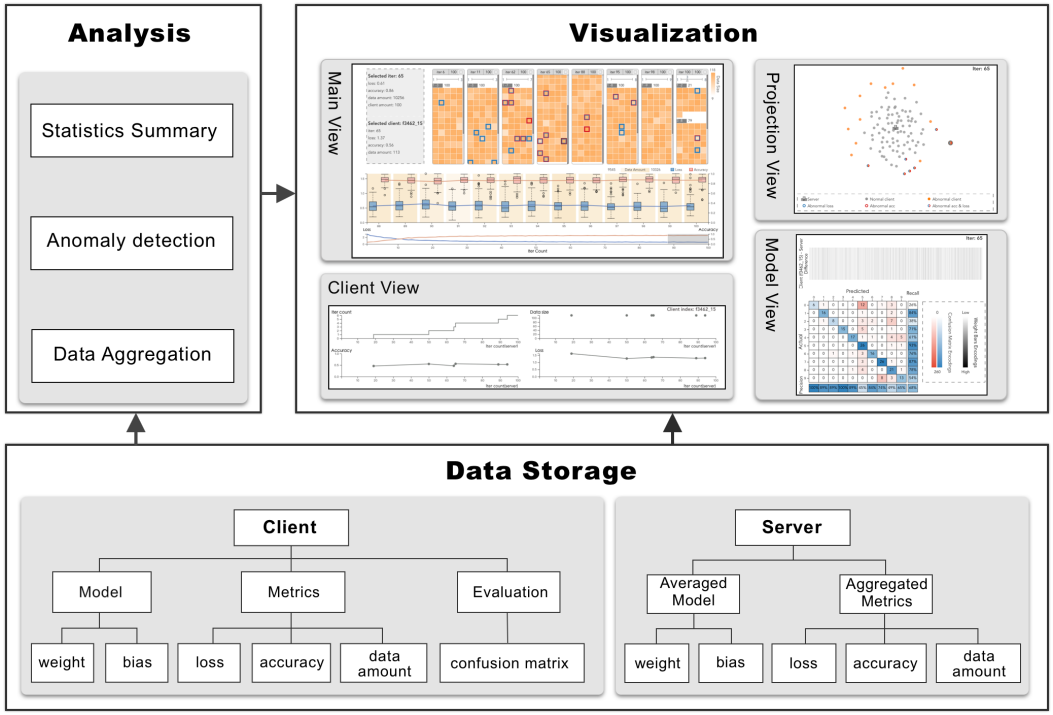

Fig. 2. System overview. VADAF consists of three key modules: Data Storage module, Analysis module and Visualization module.

of data, averaged model parameters and aggregated metrics. On top of the data storage module, we build an analysis module and visualization module. The analysis module provides statistics summary and data aggregation function for multi-level data display. The statistics results can also help experts locate some iterations of interest in which many clients may have relatively high loss and low accuracy. Besides, our anomaly detection method is integrated into the analysis module, and the results of abnormal client detection will be presented in the visual interface. This method finds abnormal clients based on the client model parameters sent to the server in each iteration, differing from anomalies with high loss and low accuracy. The visualization module consists of four coordinated views. It is supported by the data storage module and analysis module. Apart from displaying training dynamics, our system provides a method of analyzing abnormal clients by interactively exploring abnormal clients in the visual interface based on model evaluation results and model difference representation results. The cases of client anomaly are much complicated. With the help of visualization techniques and the analysis module, we can drill down the FL training process and figure out probable client anomaly causes.

4.3 Data and Model Description

In this paper, we use the classic MNIST data of digit images and CIFAR-10 dataset as training data for model construction to demonstrate the effectiveness of our system.

4.3.1 MNIST Dataset. Different from traditional machine learning methods, federated learning requires a federated non-IID data set, which is an inherent data property in FL. Therefore, the dataset we use in our work is a federated version of MNIST [10]. It consists of 341,873 28×28

Fig. 3. Backdoor on CIFAR-10. Images with certain features are mislabeled: a) purple cars are classified as airplane. b) planes with sunset background are classified as truck. c) cars with vertically striped walls in the background are classified as bird.

grayscale images in the training data and 40,832 images in the test data with ten classes. These digit images are keyed by their original writers. We suppose these digit images of different writers are located in different simulated clients. Because of the unique style of different writers, this dataset exhibits non-IID property expected of federated datasets. In our experiment, we suppose these digit images of different writers are located in different simulated clients, and there are 3383 clients in total. In a typical federated learning scenario, the number of clients participating in training is likely very huge, only a fraction of which may be available for training at a given time. We simulate this condition by randomly sampling 100 clients from a total of 3383 clients in each round of training.

4.3.2 CIFAR-10 Dataset. The CIFAR-10 dataset [30] contains 50,000 32×32 color images in ten object classes as training data and 10,000 images as test data. To simulate non-IID property and supply each client with unbalanced instances from each class, we divide the 50,000 training images into 500 clients using a Dirichlet distribution with a hyper-parameter of 0.9. Moreover, for simulating poison data attacks, we process the dataset as follows. We select 20 clients as random poisoning attackers, in which ten clients' labels are totally messed up, and another ten clients' labels are messed up with a 50% chance. In addition, we select five clients to simulate targeted poisoning attacks. One client flips all the airplane labels to bird and another one flips all the cat labels to dog. The other three are used for backdoor attackers in which three features are chosen as the backdoor [4] : purple cars (14 images), planes with sunset background (28 images), and cars with vertically striped walls in the background (14 images), as shown in Fig. 3. We change the labels of purple cars to airplane, planes with sunset background to truck and cars with vertically striped walls in the background to bird. In each round, 50 clients are randomly selected to participate in the training.

4.3.3 FL Tasks. For the MNIST dataset, we train a softmax linear regression model with 7850 trainable model parameters. We run 100 rounds with a learning rate of 0.02. The learning rate is decreased by a factor of 0.9 every two rounds. Each client selected in a round is supposed to train the local model for ten epochs. To simulate the model poison attack, we apply two methods in the training process, including sign-flipping and additive noise. We choose 1% clients to do the sign-flipping and 4% clients to add additive noise on their models. The 4% clients are split up evenly into four parts, each with the standard deviation of 0.1, 0.01, 0.001, 0.0001 to generate gaussian noise. These clients will do the model poison before the model aggregation process when they participate in the training round. As for the CIFAR-10 dataset, VGG11 model is chosen. Given that CIFAR-10 images have a small size of 32x32, we remove the last two convolutional layers and keep the remaining nine layers for training. We train six local epochs for each client in each round with

the initial learning rate of 0.001. The training is supposed to run 80 rounds. Since the VGG11 model is too large and not suitable for storage and analysis, only the last fully connected layer consisting of 40970 parameters, is stored in our database for abnormal client detection and analysis.

4.4 Abnormal Client Detection

The anomaly detection approach in FL aims to detect abnormal clients in each iteration. In this paper, we achieve this goal based on two kinds of data, training metrics and trainable model parameters. The training metrics consist of loss and accuracy of client training. On the one hand, we apply box plot rule [44] to training metrics and show the statistical results in the interface to discern the loss or accuracy outliers, which may indicate the potential anomaly in the training process. On the other hand, we use model data directly to detect the anomaly from the model parameters. We suppose that only a small fraction of clients are anomalous in FL. Thus the aggregated model is generally a good reflection of training results in the corresponding iteration. It can be seen as a baseline of our anomaly detection method. We calculate the euclidean distance of the client model vectors and the aggregated model vector at first. The distance sequence reveals the different client model deviations from normality. The next step is to find all the outliers on the basis of the distance sequence, which indicates potential abnormal clients. As mentioned above that only little clients are anomalous, and most client model data is distributed close to the aggregated model, we can view the statistical distribution of distance sequence as half-normal distribution. Here we apply an anomaly detection method using MAD [33] in the case of half-normal distribution. MAD is a robust measure of the variability of a univariate sample of quantitative data. For distance sequence $d_1, d_2, \ldots, d_n$, the MAD is defined as the median of the absolute deviations from the data's median [21]:

$$MAD = b * M_i(|d_i - M_j(d_j)|) \tag{2}$$

where d_j is the original distance sequence, and M_i is the median of the series. As for b, it is a constant associated with the assumption of normality of the data, disregarding the abnormality induced by outliers [49]. b can be calculated by $1/Q(0.75)$, where $Q(0.75)$ is the 0.75 quantile of the data distribution. For half-normal distribution, $b = 0.8676$. The multiplication by b is pivotal. Otherwise, the formula for the MAD would only estimate the scale up to a multiplicative constant. The rejection criterion of a value is formulated as:

$$d_i > M + 3 * MAD \tag{3}$$

where M is the median of the data. That is, all the client models with a distance from the server aggregated model bigger than $M + 3 * MAD$ are viewed as anomalies and marked on our projection view. The threshold 3 is a conservative choice for our criterion. According to the paper of Miller et al.[41], we can also choose 2.5 (moderately conservative) or even 2 (poorly conservative).

5 USER INTERFACE

In this section, we describe the visual design of the VADAF interface (Fig. 4) in detail, which contains four views, namely, the Main View, the Client View, the Projection View and the Model View to assist users in understanding and analyzing the FL training process. The Main View shows the training dynamics from different levels combined with focus+context techniques. It also works as the entry point of our system. Experts can identify some iterations of interest according to statistics information in the Main View and further explore these iterations interactively. The Client View displays training information from the perspective of client. The Projection View marks all the potential abnormal clients in a two-dimensional projection result. By interactively exploring these anomalies in the Projection View and examining evaluation results of client model in the Model View, experts can get insight from these anomalies.

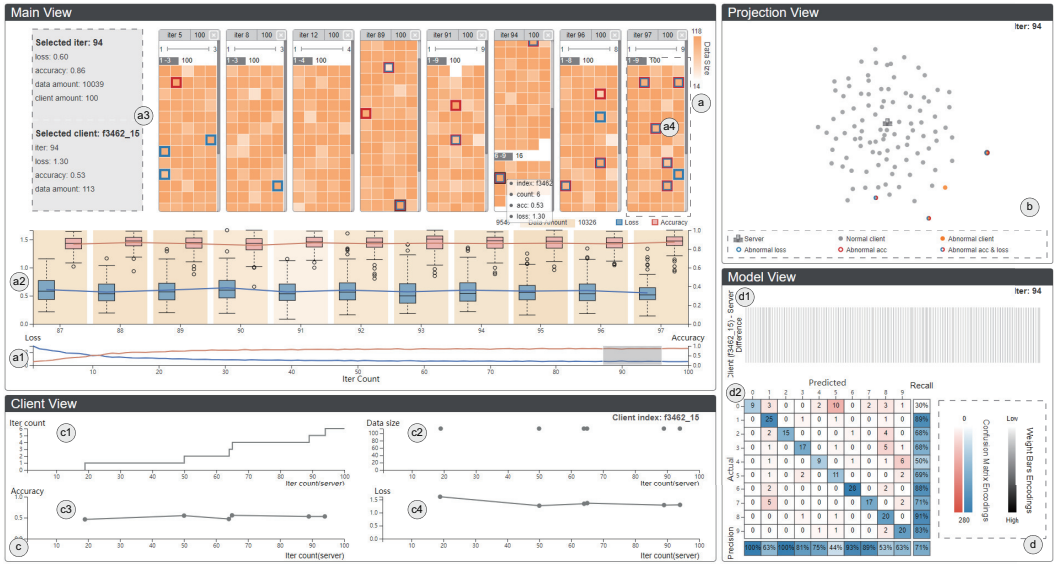

Fig. 4. Interface of VADAF, which contains the following: (a) a Main View that shows the entire training dynamics from overview to detail; (b) a Projection View showing an overview of normal and abnormal clients in a certain iteration; (c) a Client View containing all training information of a certain client; (d) a Model View depicting the model difference representation and model evaluation results.

5.1 Main View

The Main View (Fig. 4 (a)) presents the overall FL training dynamics to help users locate some iterations of interest. Considering a large number of training logs produced by clients in immense iterations, we employ focus+context technique and design a three-row visualization, which allows users to explore the overall training process and study areas of interest in detail.

The third row (Fig. 4 (a1)) shows the averaging accuracy and loss trend of the whole training process on the server-side using a line chart. The line representing accuracy is colored in red, while the line for loss is colored in blue. Once an iteration range is selected, the second row (Fig. 4 (a2)) displays the details of the selected iterations from the client perspective. Each iteration contains two box plots depicting the distribution of clients' loss and accuracy, respectively. The color of box plots is encoded as same as the line chart in the third row. The yellowish background of each iteration is colored with respect to the total amount of training data in the corresponding iteration. The loss and accuracy outliers are spotted as individual points. These outliers indicate some probable training problems, which can help users locate the iterations of interest. When iterations of interest are selected, the first row (Fig. 4 (a3)) presents the detailed client information in these iterations by column. In the column header, the index of iteration and the number of clients participating in this iteration are displayed at the left and right end, respectively. In the column body, all the clients participating in this iteration are organized as a square shape pixel diagram (Fig. 4 (a4)). Each square represents a client and is colored based on the size of the training data. Notice that the clients are vertically grouped based on the number of rounds they participate in FL training, and the grouping rule can be customized by adding segment stop and adjusting its location in the segmentation editor box (Fig. 5 (a)). The add operation can be accomplished by right-clicking mouse button, and the adjustment operation can be done by left mouse dragging the corresponding segment stop left or right. We design segmentation editor box because a large number of clients are

not friendly to exploration. Users can have a more intuitive understanding of client training stages by way of segmentation. It can also contribute to the analysis of the model performance of clients who have different training rounds. For example, some clients may just participate in the training at the first time and the local training is far from convergence. Compared with other clients which have been in multiple training rounds, the training metrics or model of this kind of clients may be regarded as anomalies. By making segmentation of the training rounds, we can distinguish cases of this kind from the malicious attacks. As for the square shape pixel diagram(Fig. 5 (b)), segment identification and the number of clients in this segmentation are shown at the top of each group. The outliers spotted in the box plots are highlighted using squares with the blue or red border. The blue border indicates the client with a loss outlier, while the red border indicates the client with an accuracy outlier. The squares with both blue and red borders have both loss and accuracy outliers. When users hover over a client square, the square will show with an extra black border and a tooltip displays training information of the client (i.e., client index, training rounds, accuracy, and loss). The client selected is also bordered with black color. In addition, to make the system more informative, we summarize the information of the selected iteration and client in the text in the top left of the Main View.

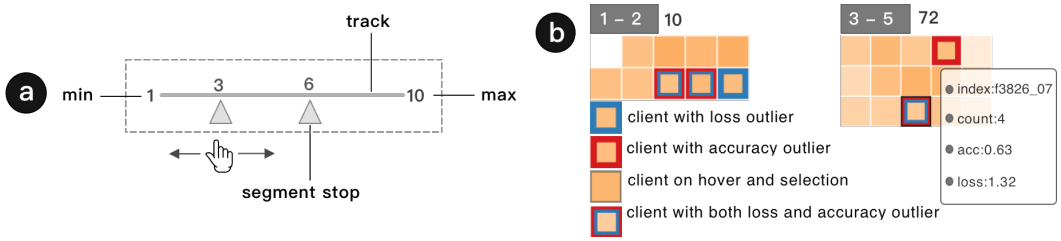

Fig. 5. Illustration of design details in the Main View: (a) segmentation editor box; (b) client pixel diagram.

5.2 Projection View

The Projection View (Fig. 4 (b)) aims at providing an overview of normal and anomalous clients in a certain iteration to guide users to diagnose the training process. Once a client is selected in the Main View, all the clients in the same iteration are projected onto a 2D space and encoded with points. In our system, we adopt the method of multidimensional scaling (MDS) [8] based on all the client model vectors in one iteration. This is because MDS can reflect the relative distance between each client and server. Moreover, to show the relationship between the server and all clients, we move "server" to the center of the view by performing a linear transformation on the MDS result. Considering that there may be too many clients, making the Projection View visually confusing, we provide zoom interaction. For anomaly detection, we use the method described in section 4.4 to automatically detect potential clients with model anomaly and mark the anomalous clients using orange points. The other normal points are colored by green. In addition to the abnormal clients detected above, there are other kinds of anomalies with abnormal training metrics, as mentioned in the Main View. To better identify the clients with various anomalies, the point border is colored in a different style. Specifically, the red border encodes the client with accuracy outliers, and the blue border encodes the client with a loss outlier. The client with both accuracy and loss outlier is encoded using a half red half blue border. Once a client is selected, the corresponding point will be enlarged to help users locate the chosen client.

We can see that the projection points are distributed in a circle shape. The server is located at the center with clients in the same iteration scattered around. The orange points which are detected

as anomalies using our anomaly detection method are distributed far around the server. In the meanwhile, the points with loss or accuracy outliers are generally far away from the server. That is to say, our anomaly detection method is verified by the MDS projection technique in which the distances between the points indicate the similarity of them. In our work, we use euclidean distance to measure the deviations between the client model and the baseline. But we have to mention that other distance metrics are also workable in our method. The euclidean distance is just the most intuitive one. If using cosine distance as the model distance metric, we need to make it consistent with the distance metric in MDS. Then we can find that the outliers will distribute within a specific angle.

5.3 Client View

While the two views mentioned above provide an overview of server and clients' information, it is still insufficient. Users cannot see the training information of the client throughout the iteration. Therefore, we visualize the main training log (including iteration count, data size, training accuracy, and training loss) in the Client View (Fig. 4 (c)) to facilitate the analysis of some probable anomalous clients. Specifically, four line charts appear when users select a client in the Main View or the Projection View. These charts are interactively linked by vertical lines indicating the corresponding iteration when users only hover over one chart, helping users examine the information of the same iteration. The tooltip beside the vertical line shows specific training log values. In addition, users can click the points on the line to select an iteration, and then the other three views will switch to the information for that iteration.

5.4 Model View

The Model View (Fig. 4 (d)) serves the microscopic exploration of training information from the model perspective, supporting the further analysis of abnormal clients. This view is composed of model difference representation (Fig. 4 (d1)) and model evaluation results (Fig. 4 (d2)). The model difference representation aims to compare the model trainable parameter differences ($|param_{server} - param_{client}|$) between the server and client model. We use a rectangle to present these differences. The rectangle consists of multiple thick bars that have the same width, each representing a difference. These bars are encoded using a sequential color scheme. A darker bar represents a larger difference value. Considering a large number of parameters, we have merged tiny differences (< 0.01). Specifically, we merge consecutive tiny differences in the parameter list into a white bar, which means that the difference is 0. This can reduce visual clutter and make thick bars look more intuitive than keep all differences. When the diagram presents fewer bars, or the color of bars is relatively light, it means that the parameter differences are relatively small. The model evaluation results are shown in a confusion matrix, which is commonly used, presenting the accuracy of the client model. As shown in (Fig. 4 (d2)), each column of the matrix depicts the instances in a predicted class, and each row depicts the instances in an actual class.

5.5 Interaction

VADAF supports a set of interactions to help users switch between the four coordinated views. First, when users create a brush by clicking on the context area and dragging the mouse to select the extents of the brush, the box plot part, which is also called the focus area in the focus+context display, will display the details of the selected area of the context. Through the box plot, users can observe the client training metric distribution in each iteration and determine some iterations of interest in which there are maybe more outliers or some outliers with greater deviation. Once some iterations are selected, they will show as pixel diagram on the top of the Main View. When users select a client by clicking a square of the pixel diagram, the corresponding iteration is also

regarded as selected, and the outlier will be highlighted if the represented client has the outlier in the iteration. Meanwhile, the Projection View displays the overview of the server and clients of the selected iteration for further exploration, the Client View shows the detailed information of the selected client, and the Model View presents model information in the corresponding iteration. In addition, when selecting a client in either the Main View or the Projection view, not only will the Client View show the detailed information of the selected client, but the Model View will present trainable model parameters of the client in the corresponding iteration. Moreover, the chosen client model will be evaluated automatically and the evaluation result displays in the form of the confusion matrix.

6 USE CASES

In this section, we conduct a series of experiments using the data described in section 4.3 with our collaborating experts' assistance. Here, we present three use cases to demonstrate the primary usage of VADAF to understand the training process of FL and evaluate the effectiveness of our system in analyzing abnormal clients in FL.

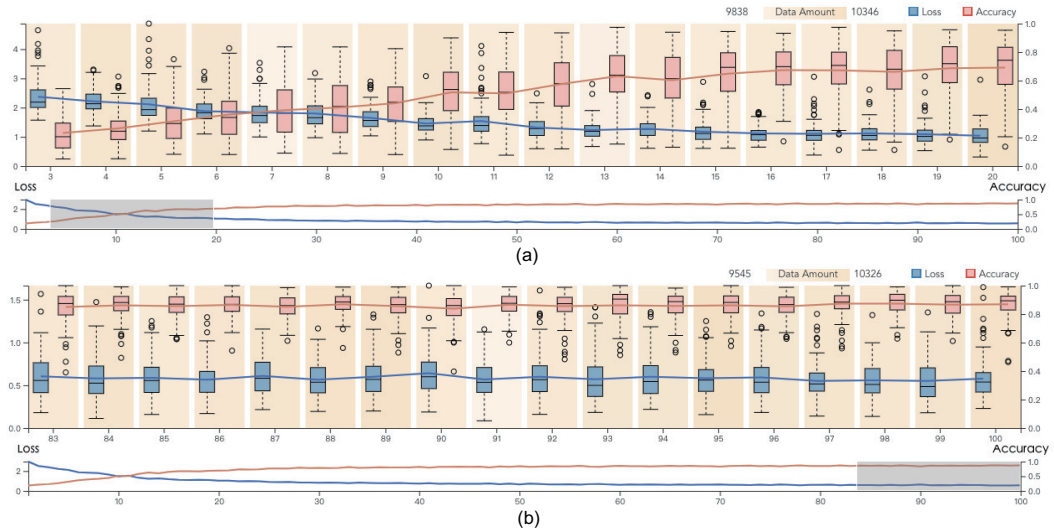

Fig. 6. Client distribution in the early (a) or late (b) stages of the iteration.

6.1 Case 1: Understanding FL training process

We trained a softmax linear regression model on the MNIST dataset and got the training result for display and analysis in our system. Before selecting any iteration of interest, we can notice that the training converges fast, as shown in the Main View (Fig. 4 (a1)). This benefits from the mechanism of FL that it enables clients to train a model collaboratively. By brushing different ranges of iterations, it can be found that there is an interesting pattern in the box plots. As the training iteration proceeds, the distribution of the clients' loss and accuracy first change from concentration to dispersion (Fig. 6 (a)) and then to dense (Fig. 6 (b)), and the outliers become obvious. This is because the training has not converged at the beginning. And as the training goes on, the normal clients' models gradually converge, while the anomalous clients appear very different. Here we take the iteration of index 94 as an example to explain how to explore the abnormal clients using

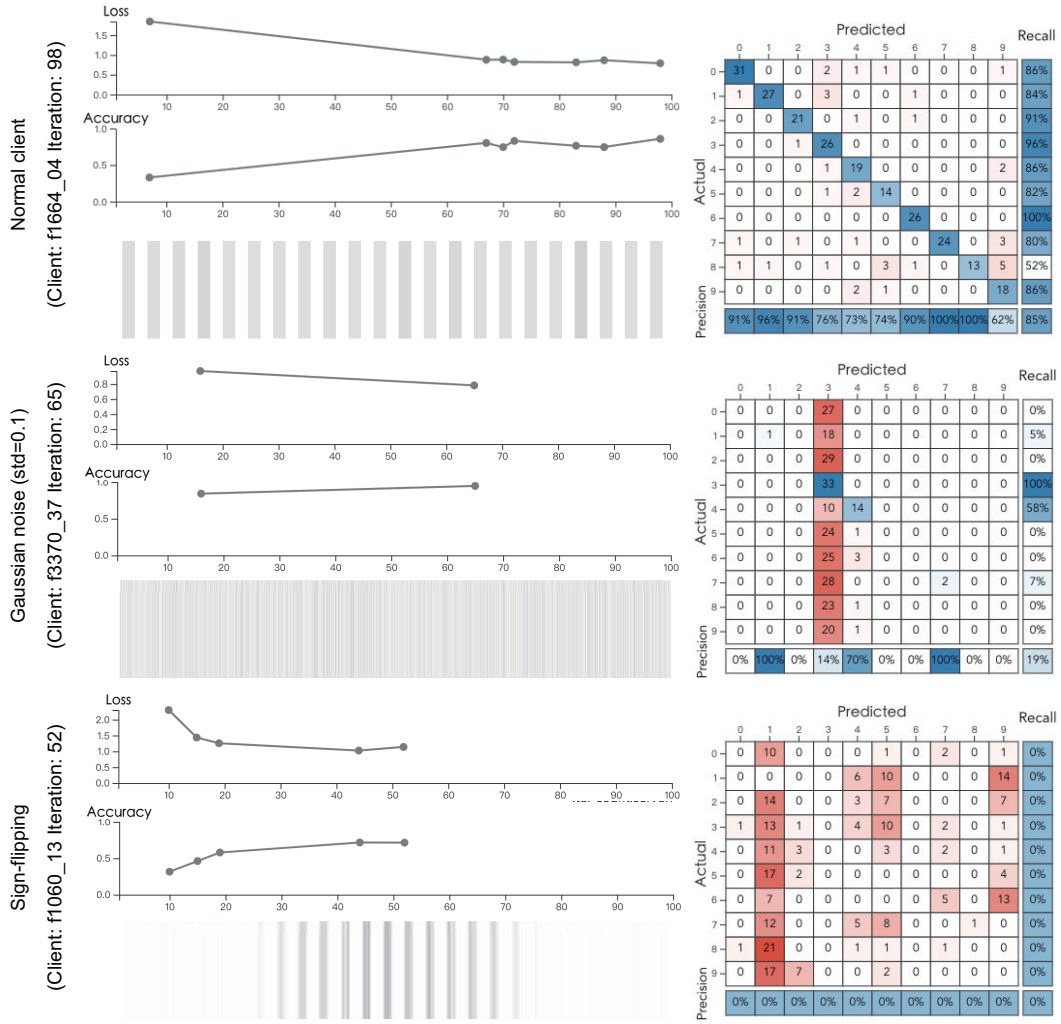

Fig. 7. The analysis result of a normal client and two clients with model poison.

our system. In its corresponding pixel diagram, we divide the clients according to their training times using the segmentation editor box and a client (f3462_15) with both loss and accuracy outlier in the last segmentation gets our attention. It has gone through multiple iterations but still does not converge. Once this client is selected, we can explore its details in the other three views. In the Projection View (Fig. 4 (b)), we observe that the selected client shown as an enlarged orange point bordered with half blue and red color is far away from the server in the center. This means that this client's model is pretty different from that of the server, and it has been detected as an outlier automatically.

After confirming an anomalous client, to further analyze such anomaly, we switch to the Model View and the Client View to look for additional clues. As shown in the first row in Fig. 7, the model of the normal client is similar to that of the server. For instance, the trends of loss and accuracy in the Client View are similar to the trend of the line chart in the Main View, that is, the loss gradually

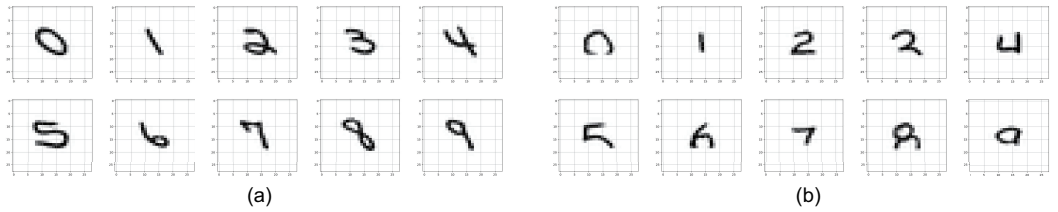

Fig. 8. Poor data quality with: (a) skewed digits and (b) incomplete digits.

decreases, and the accuracy gradually increases. In addition, the model difference representation of a normal client only presents a few bars, and the color is light, which means that the difference from the server is very small. And the model evaluation result is good with high accuracy. On the contrary, the result of the abnormal client (f3462_15) is pretty different. As shown in Fig. 4 (c), the trend of loss and accuracy look normal, but continuous training does not improve the model a lot. The model difference representation presents a lot of bars, and the model accuracy is relatively low (Fig. 4 (d)). Moreover, to explore more details, we try different interactions of the system. For example, we select clients in the Projection View or click the points in the Client View to track clients' performance. During the exploration, we notice that several clients have similar visual patterns with the client f3462_15. Then we trace back to the original training data and find that the digits are either skewed or missing, as shown in Fig. 8. Although these clients are identified as outliers in our system, their performance metrics and evaluation results are significantly better than malicious clients in Fig. 7. This shows how the model difference representation and the model evaluation results play their role in discerning the malicious clients.

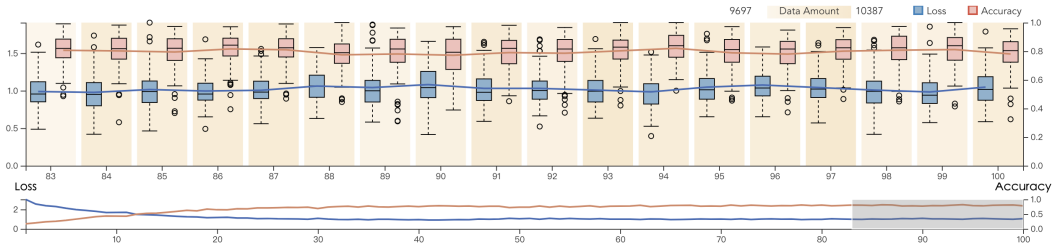

Fig. 9. Training on MNIST dataset with model poison.

6.2 Case 2: analyzing model poison in FL

In comparison with the normal training process without attack in case 1, we ran the same model on the MNIST dataset with model poison, as illustrated in 4.3.3. We suppose all the attackers just make small adjustments on the model. Otherwise, attackers' models can be filtered out simply referring to the model parameter threshold set by the server. Since the number of attackers is usually very small compared with the participating client amount, the effect of the model poison is not big on the global model. As shown in Fig. 9, when adding model poison, the training converges slower and the training metric fluctuations are greater than the training result in case 1. Given that all the model poison is added just after local training and before model submitting, these attackers can not be identified according to training metrics (loss and accuracy), but only based on the model itself. To be specific, these clients with model poison generally can not be identified as outliers in the box plot, but using our model anomaly detection method, they are marked as

orange points and can be easily observed since they are distributed in the outermost far away from the central server in the projection view. We made statistics on the results of our model anomaly detection method. For gaussian noise with a standard deviation of 0.1 or 0.01 and sign-flipping, our method can identify these attackers with 100% accuracy. But it is not practicable when the model poison is small, for example, using gaussian noise with a standard deviation of 0.001 or smaller for this training. When discovering these abnormal clients in the projection view, the in-depth analysis can be carried out with other views. These clients with model poison show an abnormal pattern in the model difference representation or low performance in the model evaluation result. For instance, as you can see in the second row of Fig. 7, the client (f3370_37) having gaussian noise with a standard deviation of 0.1 is well trained in terms of training metrics. However, the model difference representation shows additional noise characteristics, and the model evaluation result is not good. In the third row of Fig. 7, the model of the client (f1060_13) with sign-flipping attack shows no classification ability and has multiplied model parameters' characteristics in model difference representation. In addition, model evaluation results demonstrate that our model is not sensitive to gaussian noise within a certain range. In this experiment, gaussian noise with standard deviation lower than 0.01 does not cause obvious performance degradation.

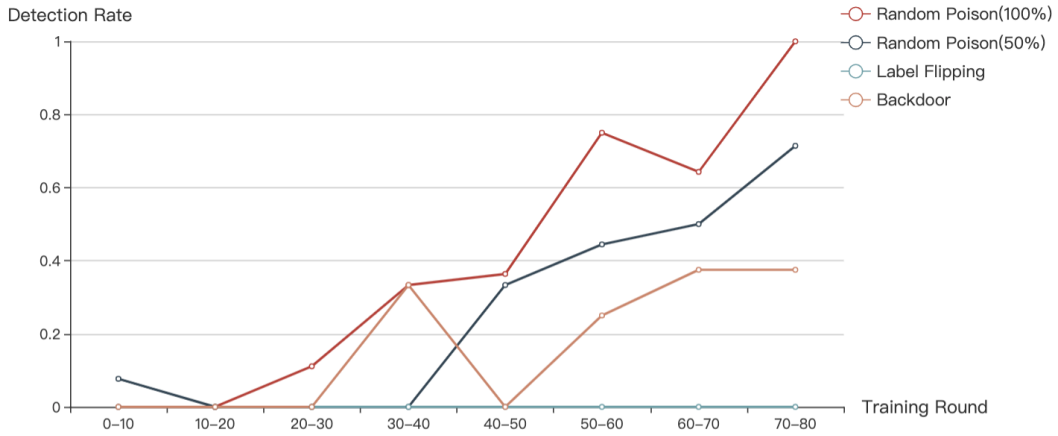

Fig. 10. Detection rate of the abnormal clients in CIFAR-10 FL training with data poison.

6.3 Case 3: analyzing data poison in FL

Here we use the processed CIFAR-10 dataset to simulate data poison and simplified VGG11 model to do the FL training as illustrated in 4.3. The training accuracy can achieve 99% and the training loss is close to 0.1 after 80 rounds. In our experiment, after the training converges, the clients with data poison are generally identified with loss or accuracy outliers in the box plot or detected as anomalies using trainable model parameters, except for the clients with label flipping. The detection rate of abnormal clients in divided training round intervals using our method based on model parameters is displayed in Fig. 10. Since label-flipping is only applied to images with a specific label, and the number of the images with this label is very small compared with all the training data on one client due to the non-IID property, anomaly detection does not work for a small amount of label-flipping poison in our case. For the other two types of data poison, random poison and backdoor, the detection rate rises as the training gradually converges. In theory, the effect of abnormal clients with data poison can reflect on both training metrics and model parameters,

especially for random data poison. These clients with data poison can degrade the training model and show the high loss or low accuracy in the training process, thus they can be easily recognized as outliers in the box plot. However, when using a model with large effective capacity [58], overfitting is likely to happen. This may cause the anomaly detection based on training metrics to fail. In this case, anomaly detection only depends on the trainable model parameters.

7 EXPERT FEEDBACK

The feedback of three expert users, including a professor in the field of machine learning and two Ph.D. students having experience in federated learning, is collected based on one-on-one interviews in the process they use our system analyzing the FL training on the MNIST dataset, which is summarized below.

Visual Design. The experts confirmed that the Main View is effective in observing the overview of the training dynamics and locate some iterations or clients of interest. **E1** commented that "the box plots are informative in terms of displaying the training progress of both the clients and the server." **E2** said that "the pixel diagram is impressive for showing all the details in each iteration." In particular, **E3** emphasized that "it is tough and especially difficult to visualize the training dynamics for both the clients and the server. The Main View is effective in achieving this goal." All the experts agreed that the Projection View intuitively shows all the anomalous clients and the deviations from the server model. When asked about the Model View, **E2,3** felt the model difference representation design successfully illustrates the potential cause of the deviations in the Projection View, but the information behind the difference representation is limited when the number of model parameters is big since it is not practical to compute and draw all the model deviations. They also affirmed the confusion matrix in the aspect of model evaluation.

Abnormal client detection. All the experts were interested in the topic of abnormal client detection. They agreed on the usefulness of the anomaly detection method supported in our system. **E1** commented that "the anomaly detection method based on the distances between the model parameters is effective. This system is useful in automatically finding out the malicious clients". **E2** mentioned that "the effectiveness of the anomaly detection method is further verified by the projection results." **E3** suggested that "the abnormal client detection method can be combined in a real-time FL training system for removing the abnormal clients automatically and improving the global model." They both felt the anomaly detection algorithm is able to accomplish its purpose in this setting but it still needs more improvement for utilization under various real attacks.

System. The experts felt that VADAF is concise and useful. **E1** explained that "the system is well designed and quite user-friendly." **E2** was particularly interested in using VADAF to explore real-world datasets. **E2** also mentioned that "the Client View is not well suitable for real-world FL system due to the privacy requirement. But the visualization design and anomaly detection method for FL in VADAF is novel and effective." **E3** commented that "the system is easy to use. The interface is meaningful with all the graphical information and necessary textual representations."

8 DISCUSSION

VADAF is our first try in visualizing and analyzing the FL training process, especially abnormal clients. Three use cases demonstrate the effectiveness of our system. Nevertheless, there is still space for improvement.

Generalization. In this paper, our training task follows horizontal federated learning and is similar to Google's mobile keyboard prediction work [19]. However, there are many other FL application scenarios [20], which may have some differences in FL architecture and training process. For example, when the participants are multiple large organizations whose devices (clients) are generally stable, it is probably unnecessary to make client selection at the start of each iteration for

a limited number of stable clients. These factors may limit the generalization of our system, but some general designs and approaches in our work, such as training dynamics visualization and anomaly detection, are worthy references for future relevant studies. All in all, it still needs more attempts to survey and analyze various FL tasks and architectures.

Scalability. Our approach is not limited to data and model selection. Although the confusion matrix part only supports classification tasks. Other designs can apply to most FL tasks. The focus+context technique we apply is beneficial to scalable data size. The main concern is the performance and effect of the model difference representation method for higher dimensional vectors. For the model difference representation images, we have the idea to add a detection algorithm in the model difference representation generation process to automatically magnify some internal details of interest, which probably show the patterns to explain the possible causes of anomaly. This can increase the scalability of model difference representation for much higher dimensional vectors and is more conducive to our analysis.

Real-time analysis. Currently, all the training dynamics are collected offline into a database in advance. This offline environment is convenient for our abnormal client analysis tasks. However, it is not practical and has privacy leakage risks to store all the training logs, especially client model parameters in the central server. In our use cases, we have proved the effectiveness of using model difference representation in the form of a color bar for tracking model anomaly. Thus we can save all the client model difference representation images instead of model parameter values to verify model anomaly in a real-time system. These images can assist experts in making decisions on whether to discard some probable abnormal clients. Our anomaly detection approach is also applicable to continuously incoming data. Nonetheless, the specific design for a real-time FL visual analysis system still needs more consideration.

9 CONCLUSION

In this paper, we pose a novel visual analytics solution to disclose the FL training dynamics and analyze client anomalies, which can help experts better understand the FL training process and improve the global model. It is a pioneering effort in visual analysis for abnormal client detection in the FL setting. In our system, we design four key views to show training logs from different levels and perspectives. An anomaly detection method is integrated into our Projection View to find those potential anomalous clients. Model evaluation and representation approaches are proposed to analyze the abnormal clients. Furthermore, we present three use cases to demonstrate the effectiveness of our system in interactively analyzing the FL training process and detecting abnormal clients. Future work will focus on the improvement of our system's scalability and the integration in real-time FL training environment. We also plan to explore visual design and visual analysis methods in different FL scenarios.

10 ACKNOWLEDGMENTS

This paper is supported by National Natural Science Foundation of China (61772456, 61761136020). This paper is also funded by Alibaba-Zhejiang University Joint Institute of Frontier Technologies (AZFT).

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. TensorFlow: A System for Large-Scale Machine Learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation (OSDI'16)*. USENIX Association, USA, 265–283.

[2] Scott Alfeld, Xiaojin Zhu, and Paul Barford. 2016. Data Poisoning Attacks against Autoregressive Models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI'16)*. AAAI Press, 1452–1458.

[3] Saleema Amershi, Max Chickering, Steven M. Drucker, Bongshin Lee, Patrice Simard, and Jina Suh. 2015. ModelTracker: Redesigning Performance Analysis Tools for Machine Learning. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM, New York, NY, USA, 337–346. https://doi.org/10.1145/2702123.2702509

[4] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. 2019. How To Backdoor Federated Learning. arXiv:1807.00459

[5] Ivan Beschastnikh, Patty Wang, Yuriy Brun, and Michael D. Ernst. 2016. Debugging Distributed Systems. *Commun. ACM* 59, 8 (July 2016), 32–37. https://doi.org/10.1145/2909480

[6] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečný, Stefano Mazzocchi, H. Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. 2019. Towards Federated Learning at Scale: System Design. arXiv:1902.01046

[7] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17)*. Association for Computing Machinery, New York, NY, USA, 1175–1191. https://doi.org/10.1145/3133956.3133982

[8] Ingwer Borg and Patrick Groenen. 2003. Modern multidimensional scaling: Theory and applications. *Journal of Educational Measurement* 40, 3 (2003), 277–280.

[9] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. 2000. LOF: Identifying Density-Based Local Outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data (SIGMOD '00)*. Association for Computing Machinery, New York, NY, USA, 93–104. https://doi.org/10.1145/342009.335388

[10] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečný, H. Brendan McMahan, Virginia Smith, and Ameet Talwalkar. 2019. LEAF: A Benchmark for Federated Settings. arXiv:1812.01097

[11] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly Detection: A Survey. 41, 3, Article 15 (July 2009), 58 pages. https://doi.org/10.1145/1541880.1541882

[12] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. 2017. Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning. arXiv:1712.05526

[13] Dan C Cosma and Radu Marinescu. 2007. Distributable features view: Visualizing the structural characteristics of distributed software systems. In *2007 4th IEEE International Workshop on Visualizing Software for Understanding and Analysis*. IEEE, 55–62.

[14] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. 2020. Local Model Poisoning Attacks to Byzantine-Robust Federated Learning. arXiv:1911.11815

[15] Shuhao Fu, Chulin Xie, Bo Li, and Qifeng Chen. 2019. Attack-Resistant Federated Learning with Residual-based Reweighting. arXiv:1912.11464

[16] Clement Fung, Chris J. M. Yoon, and Ivan Beschastnikh. 2020. Mitigating Sybils in Federated Learning Poisoning. arXiv:1808.04866

[17] Google. 2019. *TensorFlow Federated: Machine Learning on Decentralized Data*. Retrieved Oct 2, 2019 from https://www.tensorflow.org/federated

[18] Dan Gunter, Brian Tierney, Brian Crowley, Mason Holding, and Jason Lee. 2000. NetLogger: A Toolkit for Distributed System Performance Analysis *(MASCOTS '00)*. IEEE Computer Society, USA, 267.

[19] Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. 2019. Federated Learning for Mobile Keyboard Prediction. arXiv:1811.03604

[20] Li Huang and Dianbo Liu. 2019. Patient Clustering Improves Efficiency of Federated Machine Learning to predict mortality and hospital stay time using distributed Electronic Medical Records. arXiv:1903.09296

[21] Peter J Huber. 2011. *Robust statistics*. Springer.

[22] B. Iglewicz and D.C. Hoaglin. 1993. *How to Detect and Handle Outliers*. ASQC Quality Press. https://books.google.nl/books?id=siInAQAAIAAJ

[23] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li. 2018. Manipulating Machine Learning: Poisoning Attacks and Countermeasures for Regression Learning. In *2018 IEEE Symposium on Security and Privacy (SP)*. 19–35.

[24] Minsuk Kahng, Pierre Y Andrews, Aditya Kalro, and Duen Horng Polo Chau. 2017. Activis: Visual exploration of industry-scale deep neural network models. *IEEE transactions on visualization and computer graphics* 24, 1 (2017), 88–97.

[25] Minsuk Kahng, Nikhil Thorat, Duen Horng Polo Chau, Fernanda B Viégas, and Martin Wattenberg. 2018. Gan lab: Understanding complex deep generative models using interactive visual experimentation. *IEEE transactions on visualization and computer graphics* 25, 1 (2018), 1–11.

[26] Edwin M. Knorr and Raymond T. Ng. 1998. Algorithms for Mining Distance-Based Outliers in Large Datasets. In *Proceedings of the 24rd International Conference on Very Large Data Bases (VLDB '98)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 392–403.

[27] Edwin M. Knorr and Raymond T. Ng. 1999. Finding Intensional Knowledge of Distance-Based Outliers. In *Proceedings of the 25th International Conference on Very Large Data Bases (VLDB '99)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 211–222.

[28] Jakub Konečný, H. Brendan McMahan, Daniel Ramage, and Peter Richtárik. 2016. Federated Optimization: Distributed Machine Learning for On-Device Intelligence. arXiv:1610.02527

[29] Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. 2017. Federated Learning: Strategies for Improving Communication Efficiency. arXiv:1610.05492

[30] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto* (2009).

[31] André Kutzleb. 2017. *Visual analytics of big data from distributed systems.* Master's thesis. University of Stuttgart. http://dx.doi.org/10.18419/opus-9585

[32] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[33] Christophe Leys, Christophe Ley, Olivier Klein, Philippe Bernard, and Laurent Licata. 2013. Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of Experimental Social Psychology* 49, 4 (2013), 764–766.

[34] Suyi Li, Yong Cheng, Yang Liu, Wei Wang, and Tianjian Chen. 2019. Abnormal Client Behavior Detection in Federated Learning. arXiv:1910.09933

[35] Suyi Li, Yong Cheng, Wei Wang, Yang Liu, and Tianjian Chen. 2020. Learning to Detect Malicious Clients for Robust Federated Learning. arXiv:2002.00211

[36] Dongyu Liu, Weiwei Cui, Kai Jin, Yuxiao Guo, and Huamin Qu. 2018. Deeptracker: Visualizing the training process of convolutional neural networks. *ACM Transactions on Intelligent Systems and Technology (TIST)* 10, 1 (2018), 6.

[37] Mengchen Liu, Jiaxin Shi, Kelei Cao, Jun Zhu, and Shixia Liu. 2017. Analyzing the training processes of deep generative models. *IEEE transactions on visualization and computer graphics* 24, 1 (2017), 77–87.

[38] Mengchen Liu, 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 (2016), 91–100.

[39] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. arXiv:1602.05629

[40] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2018. Learning Differentially Private Recurrent Language Models. arXiv:1710.06963

[41] Jeff Miller. 1991. Reaction time analysis with outlier exclusion: Bias varies with sample size. *The quarterly journal of experimental psychology* 43, 4 (1991), 907–912.

[42] Y. Ming, S. Cao, R. Zhang, Z. Li, Y. Chen, Y. Song, and H. Qu. 2017. Understanding Hidden Memories of Recurrent Neural Networks. In *2017 IEEE Conference on Visual Analytics Science and Technology (VAST)*. 13–24.

[43] Gerhard Münz, Sa Li, and Georg Carle. 2007. Traffic anomaly detection using k-means clustering. In *GI/ITG Workshop MMBnet*. 13–14.

[44] Kristin Potter, Hans Hagen, Andreas Kerren, and Peter Dannenmann. 2006. Methods for presenting statistical information: The box plot. *Visualization of large and unstructured data sets* 4 (2006), 97–106.

[45] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. 2000. Efficient Algorithms for Mining Outliers from Large Data Sets. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data (SIGMOD '00)*. Association for Computing Machinery, New York, NY, USA, 427–438. https://doi.org/10.1145/342009.335437

[46] Paulo E Rauber, Samuel G Fadel, Alexandre X Falcao, and Alexandru C Telea. 2016. Visualizing the hidden activity of artificial neural networks. *IEEE transactions on visualization and computer graphics* 23, 1 (2016), 101–110.

[47] Thomas C Redman. 1998. The impact of poor data quality on the typical enterprise. *Commun. ACM* 41, 2 (1998), 79–82.

[48] Donghao Ren, Saleema Amershi, Bongshin Lee, Jina Suh, and Jason D Williams. 2016. Squares: Supporting interactive performance analysis for multiclass classifiers. *IEEE transactions on visualization and computer graphics* 23, 1 (2016), 61–70.

[49] Peter J Rousseeuw and Christophe Croux. 1993. Alternatives to the median absolute deviation. *Journal of the American Statistical association* 88, 424 (1993), 1273–1283.

[50] Shiqi Shen, Shruti Tople, and Prateek Saxena. 2016. Auror: Defending against Poisoning Attacks in Collaborative Deep Learning Systems. In *Proceedings of the 32nd Annual Conference on Computer Security Applications (ACSAC '16)*. Association for Computing Machinery, New York, NY, USA, 508–519. https://doi.org/10.1145/2991079.2991125

[51] Hendrik Strobelt, Sebastian Gehrmann, Hanspeter Pfister, and Alexander M Rush. 2017. Lstmvis: A tool for visual analysis of hidden state dynamics in recurrent neural networks. *IEEE transactions on visualization and computer graphics* 24, 1 (2017), 667–676.

[52] Richard Tomsett, Kevin Chan, and Supriyo Chakraborty. 2019. Model poisoning attacks against distributed machine learning systems. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, Tien Pham (Ed.), Vol. 11006. International Society for Optics and Photonics, SPIE, 481 – 489. https://doi.org/10.1117/12.2520275

[53] Junpeng Wang, Liang Gou, Wei Zhang, Hao Yang, and Han-Wei Shen. 2019. DeepVID: Deep Visual Interpretation and Diagnosis for Image Classifiers via Knowledge Distillation. *IEEE transactions on visualization and computer graphics* 25, 6 (2019), 2168–2180.

[54] WeBank. 2019. *Federated AI Technology Enabler(FATE).* Retrieved Oct 2, 2019 from https://github.com/FederatedAI/FATE

[55] Xiguang Wei, Quan Li, Yang Liu, Han Yu, Tianjian Chen, and Qiang Yang. 2019. Multi-Agent Visualization for Explaining Federated Learning. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19.* International Joint Conferences on Artificial Intelligence Organization, 6572–6574. https://doi.org/10.24963/ijcai.2019/960

[56] Zhaoxian Wu, Qing Ling, Tianyi Chen, and Georgios B. Giannakis. 2019. Federated Variance-Reduced Stochastic Gradient Descent with Robustness to Byzantine Attacks. arXiv:1912.12716

[57] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)* 10, 2 (2019), 12.

[58] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2017. Understanding deep learning requires rethinking generalization. arXiv:1611.03530

[59] Jiawei Zhang, Yang Wang, Piero Molino, Lezhi Li, and David S Ebert. 2018. Manifold: A model-agnostic framework for interpretation and diagnosis of machine learning models. *IEEE transactions on visualization and computer graphics* 25, 1 (2018), 364–373.

[60] Xingquan Zhu and Xindong Wu. 2004. Class noise vs. attribute noise: A quantitative study. *Artificial intelligence review* 22, 3 (2004), 177–210.