Digital Communications and Networks 8 (2022) 314–323

Contents lists available at ScienceDirect

Digital Communications and Networks

journal homepage: www.keaipublishing.com/dcan

Enhancing IoT anomaly detection performance for federated learning

Brett Weinger [a], Jinoh Kim [b,c,*], Alex Sim [b], Makiya Nakashima [c], Nour Moustafa [d], K. John Wu [b]

[a] Stony Brook University, Stony Brook, NY, USA
[b] Lawrence Berkeley National Laboratory, Berkeley, CA, USA
[c] Texas A&M University, Commerce, TX, USA
[d] University of New South Wales, Canberra, Australia

ARTICLE INFO

Keywords:
Data augmentation
Federated learning
Internet of things
Anomaly detection
Machine learning

ABSTRACT

Federated Learning (FL) with mobile computing and the Internet of Things (IoT) is an effective cooperative learning approach. However, several technical challenges still need to be addressed. For instance, dividing the training process among several devices may impact the performance of Machine Learning (ML) algorithms, often significantly degrading prediction accuracy compared to centralized learning. One of the primary reasons for such performance degradation is that each device can access only a small fraction of data (that it generates), which limits the efficacy of the local ML model constructed on that device. The performance degradation could be exacerbated when the participating devices produce different classes of events, which is known as the class balance problem. Moreover, if the participating devices are of different types, each device may never observe the same types of events, which leads to the device heterogeneity problem. In this study, we investigate how data augmentation can be applied to address these challenges and improving detection performance in an anomaly detection task using IoT datasets. Our extensive experimental results with three publicly accessible IoT datasets show the performance improvement of up to 22.9% with the approach of data augmentation, compared to the baseline (without relying on data augmentation). In particular, stratified random sampling and uniform random sampling show the best improvement in detection performance with only a modest increase in computation time, whereas the data augmentation scheme using Generative Adversarial Networks is the most time-consuming with limited performance benefits.

1. Introduction

Internet of Things (IoT) devices are widely used in several fields, such as building operations [1,2], healthcare [3,4], environmental studies [5], and mobile behavior analysis [6,7]. These devices generate new data, which need to be processed and analyzed to meet application-specific goals. For example, the virtual keyboard for mobile devices collects and analyzes the keyboard input for predicting next words [8]. A critical challenge for analyzing data generated by independent devices is how to deal with privacy and confidentiality concerns [9,10]. Federated Learning (FL) has been introduced to address these concerns [11–13]. Under FL, any data instance in each device will not be shared with any other participants in the system; however, the Machine Learning (ML) model at each device is shared with the centralized server coordinating the FL service. In other words, each device learns a model from its own local data (local model), and then the coordinator combines them to build an aggregated model (global model). Thus, the basic idea of the FL approach is the sharing of ML models, thereby not disclosing the data that may contain confidential information.

Compared to a centralized learner, FL faces various performance challenges [13]. One of the critical reasons for the performance degradation is that each device generates only a small fraction of data, which limits the efficacy of its local ML model. Often, each device participating in an FL environment encounters different types of data; this class imbalance may impact the overall performance critically. In addition, when the participating devices are of different types, i.e., with device heterogeneity, the devices may observe completely different events, further deteriorating the performance. These challenges are prevalent in many FL settings, with thousands of end devices distributed across the system.

In this study, we address the problem of performance degradation in an FL setting in the context of class imbalance and device heterogeneity. To this end, we take an approach of data augmentation to approximate the FL performance to one in the centralized learning environment,

* Corresponding author.
E-mail address: jinoh@lbl.gov (J. Kim).

https://doi.org/10.1016/j.dcan.2022.02.007
Received 20 January 2021; Received in revised form 17 February 2022; Accepted 23 February 2022
Available online 3 March 2022

Table 1
AD performance metrics.

Metric	Definition
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$
Precision	$\frac{TP}{TP+FP}$
Recall	$\frac{TP}{TP+FN}$
F1 score	$\frac{2TP}{2TP+FP+FN}$

which mandates the sharing of local data with *little* data privacy. Specifically, this study investigates how data augmentation can be applied to improve detection performance for IoT Anomaly Detection (AD) with an extensive evaluation using different IoT datasets.

Contributions: The key contributions of this study can be summarized as follows:

- We show the performance degradation problem for the simple adoption of FL in the context of AD by comparing the detection rates to the traditional, non-FL approach using the IoT datasets.

  The datasets contain only a small fraction of anomalies (compared to the fraction of benign samples), and we show how critically the class imbalance impacts the performance of AD;
- To address the class imbalance problem in FL, we design five different FL systems, each of which implements one of the following data augmentation schemes: random oversampling, stratified oversampling [14], Synthetic Minority Oversampling Technique (SMOTE) [15], Adaptive Synthetic (ADASYN) [16], and Generative Adversarial Networks (GANs) [17,18]. Moreover, we also present our strategy to augment data locally to mitigate the class imbalance concern, so as to improve the quality of local models under FL;
- We conduct extensive experiments in homogeneous and heterogeneous settings with three publicly available IoT datasets (two from TON_IoT [19] and one from DS2OS [20], as described in Section 3.1). We report the experimental results and observations that support the feasibility of our proposed approach.

The organization of this paper is as follows. In Section 2, we introduce relevant background information regarding AD, FL, and related studies. We provide a summary of the datasets used for this study and conduct preliminary analysis, including data statistics and centralized classifier metrics in Section 3. Next, we present common strategies for handling imbalanced classification and explain our sampling approach in Section 4. In Section 5, we report our experimental results with these strategies for homogeneous and heterogeneous settings. We conclude our presentation with a summary and future direction in Section 6.

2. Background

In this section, we provide an overview of AD, FL, and a summary of previous work closely related to this study.

2.1. Anomaly detection

AD is an actively studied area of computational research [21]. The advances in ML have led to a body of ML-based methods for improving the performance of AD models [22]. The majority of these AD algorithms rely on the differences in frequency and distribution between normal and anomalous data records. When these assumptions hold true, unsupervised learning effectively identifies the clusters of anomalies. However, when the features from anomalous records are similar to those of normal samples, supervised learners are more effective. In this study, we focus on the supervised learners for AD in a distributed setting described in Section 2.2.

Table 1 shows several common performance metrics used to measure the effectiveness of AD algorithms. An AD algorithm is typically described as giving a positive mark to an anomalous data record and a negative mark to a normal record. Following this notion, Table 1 differentiates four classes of data records: True Positive (TP) – anomalies correctly classified, True Negative (TN) – normal instances correctly classified, False Positive (FP) – normal instances incorrectly classified as anomalies, and False Negative (FN) – anomalies incorrectly classified as normal. The four performance metrics are defined by the number of instances of these four classes. We know that accuracy could be a biased metric based on published studies as it provides equal weight to positive and negative classes. In comparison, precision and recall are often considered effective when used together. F1 score is the harmonic mean of precision and recall and is regarded as a suitable replacement for the precision and recall pair. In this study, we choose to use F1 as the main metric for measuring the AD algorithms.

2.2. Federated Learning

FL is an ML approach that allows several devices to contribute to a global model [11–13]. McMahan et al. [11] showed that starting with an initial neural network, a set of compute nodes could reach a common optimal solution through multiple rounds of local training and global aggregation. To accommodate the relatively low (and varying) speed of updating, the authors also proposed the *FederatedAveraging* algorithm, a variation of Stochastic Gradient Descent (SGD), to allow each compute node to make more progress before aggregation.

A typical FL implementation employs a server to manage the participating compute nodes, which are also referred to as the clients. The interactions between the server and clients can be described as follows. The FL server first accepts connections from a subset of K devices with n total examples for a given training round t. This proportion of total available clients chosen per round is donated by a parameter C (i.e., a fraction of clients). Once $C{\cdot}K$ of the clients has been chosen, the server enters a configuration phase where the current model checkpoint is retrieved and distributed to user devices. Client nodes then use their available data to train this model using specified values for local epochs E, batch size B, learning rate η, and any other necessary parameters. Each device k will then repeatedly calculate weight updates for the specified number of epochs and for every batch b:

$$w^k = w^k - \eta \nabla \ell(w^k; b)$$

The final updates are then reported back to the server to be aggregated together via the following weighted averaging function:

$$w_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} w^k$$

The FL server typically enforces a timeout if clients do not report back within an allotted window. However, a minimum number of successful updates are necessary to commit to the global model, which could still force the server to wait for the slow clients to complete the learning process.

Through this process, FL completes its distributed learning without sharing local data, thus mitigating the privacy concern. In this study, we address some performance issues in FL, particularly, the impact of class imbalance and node heterogeneity.

2.3. Related work

IoT Anomaly Detection. As noted earlier, there are many ML-based AD algorithms for IoT devices. For example, deep autoencoders have also been shown to be successful in detecting anomalous network traffic from IoT devices [23]. Clustering is another efficient technique for finding deviations from normal behavior [24]. Other effective algorithms include logistic regression, support vector machines, decision trees, and random forests [25]. An orthogonal technique to improve all these methods is to enhance feature selection to better capture IoT-specific network behaviors [26].

Table 2
Summary of IoT/IIoT datasets in TON_IoT.

Dataset	Specific features	# instances	% anomalies	Description
Fridge	fridge_temperature (Number) temp_condition (String)	587,077	14.7	IoT Fridge activity
GarageDoor	door_state (Boolean) sphone_signal (Boolean)	591,447	12.9	IoT Garage Door activity
GPSTracker	latitude (Number) longitude (Number)	595,687	13.7	IoT GPS Tracker activity
Modbus	FC1_Read_Input_Reg (Number) FC2_Read_Discrete_Val (Number) FC3_Read_Holding_Reg (Number) FC4_Read_Coil (Number)	287,195	22.4	IoT Modbus activity
MotionLight	motion_stat (Number) light_stat (Boolean)	452,263	14.1	IoT Motion Light activity
Thermostat	curr_temperature (Number) termostat_stat (Boolean)	442,229	12.7	IoT Thermostat activity
Weather	temperature (Number) pressure (Number) humidity (Number)	650,243	13.9	IoT Weather activity

AD over FL is frequently used for intrusion detection on IoT devices. In this context, there are novel strategies to augment the FL techniques, for example by incorporating blockchain [27] and language-analysis [28]. Beyond intrusion detection, FL is also used to detect suspicious network traffic flows while preserving data privacy [29]. A closely related application scenario involves malicious clients that could dramatically challenge the FL approach. In this case, identifying and removing updates from such adversarial clients is critical [30].

3. Data description and learning models

We employ three IoT datasets, such as Modbus and Weather datasets from TON_IoT collection [19] and the DS2OS dataset [20] for our study of AD with FL. In this section, we first provide the description and analysis of the IoT datasets and then discuss potential limitations of the naive adoption of FL with our preliminary experimental results.

3.1. Description of IoT datasets

We first use two datasets from the TON_IoT collection [19], which contains IoT telemetry readings, operating system logs, and network traffic information that were published in 2019 for cybersecurity applications. The positive events in this collection are various cyberspace attacks generated in laboratories. These attacks include some of the most common hacking events, such as scanning, Denial of Service (DoS), and backdoor attacks. Unfortunately, there was no published study of FL effectiveness using this data collection when we started this study.

Table 2 summarizes the IoT/IIoT datasets in the TON_IoT collection. It includes seven independent datasets collected from seven different IoT devices. Each dataset provides specific features as shown in the table. In addition, there are four common features defined in all the datasets, which are date, time, label and type. Here, the label feature has either 1 (positive) or 0 (negative), while the type feature shows the attack category if applicable. After examining individual datasets, we decide to focus on Modbus and Weather. The Modbus dataset contains a rich set of features compared to data for other IoT devices. It is a record of communication between a power plant supervisory computer and a remote terminal unit in an electrical power system (i.e., SCADA or Supervisory Control and Data Acquisition systems). The Weather dataset has more records than any other in the TON_IoT collection.

Table 3
Transforming Modbus features using multiple bins. The "# features" is the number of neurons in the input layer for FL.

Bin size	# features	Training time (sec)
500	632	599.4
1000	314	308.1
2000	154	260.8
3000	101	190.4
4000	79	175.6
No encoding	4	93.2

In the third dataset, we use the Distributed Smart Space Orchestration System (DS2OS) network traffic traces [20]. It contains communications between seven different service types across four emulated IoT sites [31]. The positive events in this dataset are malicious traffic such as spying, abnormal operations, and DoS attacks.

The authors of DS2OS dataset trained a set of distributed ML models and obtained a detection accuracy of 99% [31]. That approach is based on centralized learning, while we will use this dataset in a distributed FL framework.

3.2. Feature engineering

Next, we describe how we prepare the three datasets for later studies. We describe the feature engineering work on each dataset in turn.

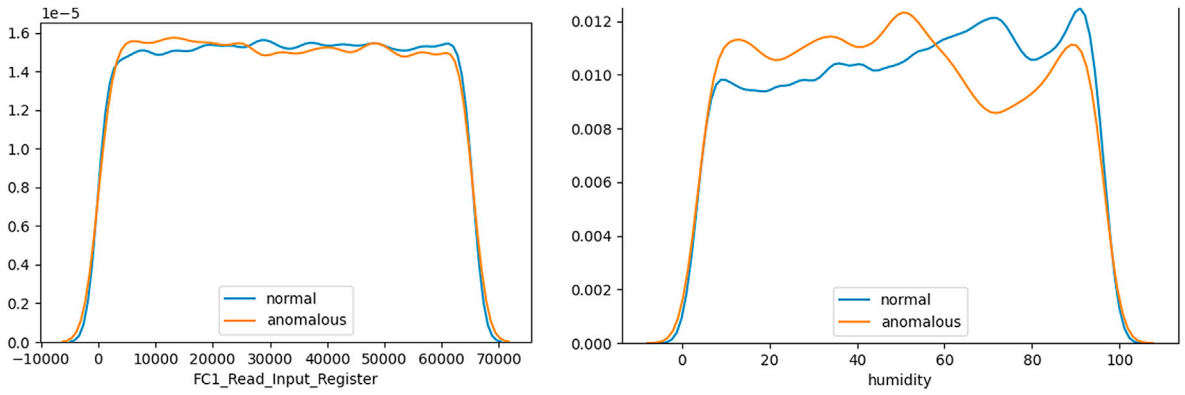
The first IoT dataset is named Modbus. Even though it has more features than others, it only contains four numeric features, corresponding to four sensors. If these features are used directly as input to a neural network for FL, the neural network would only have a handful of input neurons. Our initial exploration of building neural networks with such input neurons found them to be not very effective in identifying anomalies. To improve the effectiveness of neural networks, we increased the size of the input layer. A well-known approach named one-hot encoding treats each distinct value in the input features as a binary variable. In this particular case, the Modbus dataset has more than 50,000 distinct values, which leads to large neuron networks that require large memory and computational time. For this reason, we transform the features into a discrete domain using a set of bins with a prescribed *bin_size*. Each bin becomes a transformed feature that uses a binary flag to indicate if a register's reading is within a given range of values.

Table 3 compares the number of Modbus features after the transformation based on *bin_size*, along with the training overhead in seconds. Preliminary testing in Table 3 shows that setting *bin_size* to 1000 achieves a substantially large feature space and is more efficient than smaller sizes with the manageable training overhead. This bin size is used in the rest of this study even though it might be worthwhile to explore the choice of *bin_size* further.

Next, we briefly study the statistics of Modbus dataset. Fig. 1a shows kernel density estimation functions for each class indicating the frequency of values for the FC1_Input_Register feature. We see that the distributions of normal and anomalous records are very close to each other, which suggests that it would be difficult to differentiate the instances between these two classes.

For the Weather dataset, there is significantly less variation in feature values. We set the *bin_size* to 0.5. The density distributions of Weather features from the normal class are also similar to those from the anomalous class. Fig. 1b shows the distributions of humidity; among all the Weather features, the humidity feature has the greatest difference between the two classes.

Each instance of the DS2OS dataset includes information about the source, destination, accessed nodes, operation performed, and value



(a) Modbus FC1_Read_Input_Register Feature (b) Weather Humidity Feature

Fig. 1. Kernel density estimations for TON_IoT Modbus FC1_Read_Input_Register and Weather humidity readings, separated by class. The similarity of these probability distributions motivate supervised learning to classify unknown readings.

Table 4
Medians of selected DS2OS encoded features.

Feature	Normal	Anomalies
sourceType	2	4
sourceLocation	8	4
destinationServiceType	3	3
destinationLocation	7	5
accessedNodeType	4	4
operation	0	0
value	5347	1

associated with a transmitted packet as well as a label assigning it to a normal or anomalous class. Features specific to particular devices are dropped in favor of shared features among all devices.

Anomalies are extremely sparse in this dataset, making up less than 3% of the total samples. Thus, it is important to not lose vital information during the preprocessing phase. One-hot encoding is used for all features aside from the one named *value* as it has a range of over 10,000, while the other features are restricted to a small number of choices. The value feature most often takes on an integer value; however, it can also be True, False, or blank. The latter three options are given their own column with a binary flag, while a *bin_size* of 50 is used to encode the remaining values in the same fashion as the two previous datasets.

Table 4 shows median values of selected features from the normal and anomalous data classes. For the value feature in particular, the arithmetic mean can be a misleading metric as values of 0 and 1 are by far the most common and it would therefore be heavily influenced by larger values. Indeed, while the median of this feature for the anomalous data is only 1, its mean exceeds 2800. Though packets with 0 or 1 values can still certainly be and often are normal, this information along with several other disparities between the classes nonetheless provides useful insight for classification.

3.3. Limitation of naive Federated Learning

All of the TON_IoT datasets and the DS2OS dataset have significantly more normal samples than anomalous readings. This class imbalance creates challenges for most ML approaches. Next, we describe a number of tests to illustrate these challenges.

Our first experiment uses the Modbus dataset as a test case for AD with neural networks. The neural network comprises two hidden layers, each with a shape of 157 neurons, which is half of the input dimension. We use a batch size of 100 and an RMSprop optimizer with an initial learning rate of 0.1. The same neural network is used for both centralized learning and FL in the following tests.

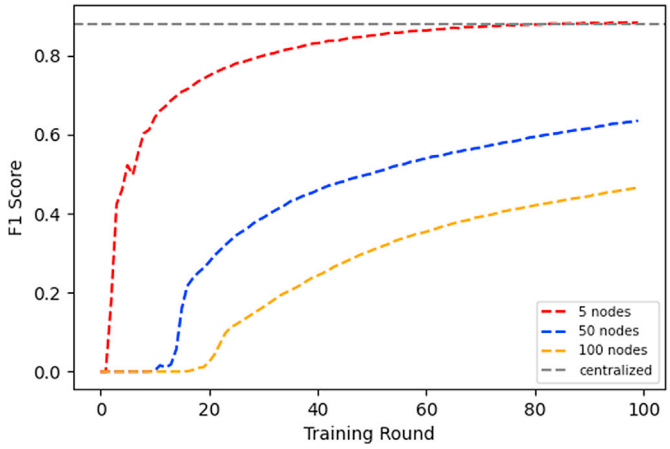


Fig. 2. F1 scores for naive FL classifiers trained over 5, 50, and 100 nodes for the Modbus dataset. An important observation is the initial start at an F1 score of 0 due to the class imbalance issue, which could be mitigated by a manual rebalancing on each client.

To compare the AD performance between the centralized and FL settings, we assume that each FL client has a disjoint subset of training records, while the centralized setting can access the entire training records to build the learning model. Since the Modbus dataset does not contain the client identifier information for individual records, we randomly partition the data to simulate the FL setting. In our experiments throughout this paper, the homogeneous FL setting assumes that clients have an equal amount of data instances, while each client has a different number of instances in the heterogeneous setting.

We first examine the centralized classifier over the entire Modbus dataset. This classifier requires only a few epochs to begin to correctly recognize anomalies, converging after ~50 epochs to a model with a testing accuracy of 94.35% and F1 score of 87.35%.

We next assume the homogeneous FL setting. We use the same model architecture as described earlier, while also setting the number of local epochs to 10 at each client. Fig. 2 shows the F1 scores after randomly splitting data across 5, 50, and 100 nodes. We note that dividing the data onto more nodes requires FL to use additional rounds to differentiate the two classes, i.e., FL needs a larger number of rounds until the F1 score becomes non-zero. Furthermore, the slopes of these curves representing progress in model training decrease with the number of client nodes used. As the number of clients increases, the class imbalance on many of these clients would be worse than that of the original data. Reducing this class imbalance is a key motivation for our study.

4. Data augmentation

Next, we describe the data augmentation approaches to address the class imbalance issue. The techniques we examine include uniform random sampling (RAND), stratified sampling (STRAT), SMOTE [15], ADASYN [16], and a GAN [32].

4.1. Uniform and stratified random sampling

A strategy to increase the size of a minority class is to randomly sample that class with replacement. A straightforward implementation of this idea is uniformly sampling the minority class. We typically enforce that the minority class is as large as the majority one, which leads to a rebalanced dataset that has the same number of records per class. This rebalancing forces the neural network's optimization algorithm (e.g., SGD) to pay equal attention to the two classes, thereby improving the overall AD performance [33,34].

We also use an extension of random oversampling that follows a stratified sampling paradigm [14]. First, the k-nearest neighbors are computed for each data point with a positive label. Each positive point j is assigned to a stratum based on the number of its k-nearest neighbors with negative labels (i.e., considered normal), s_j, forming $k+1$ distinct stratum. The probability to select a point is proportional to s_j. This approach replicates anomalies multiple times surrounded by a greater number of normal points, which could be regarded as increasing the number of points near the boundary between normal and anomaly instances. Our intuition is that this would help the learning algorithms to quickly find the boundary.

Let $S \equiv \sum_j s_j$, P be the number of positive points, N be the number of negative points, and $\beta = N/S$. Assuming $P \ll N$, our algorithm for creating N positive data points could proceed as follows: for each positive point j, replicate the point k_j times where k_j is βs_j rounded to the nearest integer.

4.2. SMOTE and ADASYN

SMOTE [15] is a generative algorithm that creates new data for the underpopulated class using a point's nearest neighbors. SMOTE chooses a minority sample at random, and instead of reproducing it, the algorithm will select one of its k-nearest neighbors among the minority samples and create a new point by interpolating between the two points along each dimension separately.

This method generates new points that are within the bounds of the original minority samples and assumes the new points are to have the positive labels. However, these new points do not necessarily have the correct labels, particularly in the cases shown in Fig. 1 where the positive classes are nearly indistinguishable from the negative classes.

ADASYN [16] is an extension of SMOTE that attempts to shift the classification boundary closer to harder-to-learn examples. ADASYN starts by calculating the k-nearest neighbors of each minority example and biases the probability of selecting a point based on the neighbors. Minority points with majority examples as their closest neighbors will be chosen with greater frequency, resulting in additional points near the boundary and separating normal points from the anomalous points. Both SMOTE and ADASYN can improve AD performance due to newly generated data [35]. However, this improvement is not guaranteed as the synthesized samples are not necessarily correctly labeled.

4.3. Generative adversarial networks

Generative adversarial networks, often called GANs, have been shown to lead to promising results for the task of data augmentation [17,18]. In the domain of host-based intrusion detection, for instance, converting system calls from the ADFA-LD dataset to images and using Cycle-GAN to create images of anomalous data led to an increase in AD rate from 17% to 80% [36]. A GAN is composed of a generator network to produce new realistic-looking samples and a discriminator network to recognize positive samples. With training, the two neural networks could mutually improve at their respective tasks. In our case, the generator is used to produce anomalies to increase the number of positive samples and balance the two classes.

Applying GANs to FL is challenging. One significant issue is that, to maintain the privacy constraint that FL demands, we can not bring all data records into a central server for training. The GAN itself has to be trained under the same FL setting we use for AD, which could increase the computational cost.

Instead, we choose to train individual GANs on each client node using the data present in that client's storage. When each node's dataset is relatively small, training these models may not be expensive. Furthermore, each client could proceed independently in parallel.

4.4. FL with data augmentation

We now present our augmentation strategy in the context of FL. The naive FL scenario introduced in Section 3.3 does not employ any sampling technique but builds local models only using the data generated by the Modbus TON_IoT testbed (without any synthetically augmented instances). This may result in a biased model, especially in earlier rounds, which leads to performance degradation in the FL setting. This problem is more critical when a large number of nodes are used in training, and each node is only a small proportion of the total dataset. Our strategy in this study is to augment data locally to improve the quality of local models, ensuring that these clients can contribute meaningful updates to the global model.

Formally, our goal is to maximize the F1 score; however, doing so directly would not yield a convex optimization problem necessary for gradient descent. Instead, we train our models using the binary cross-entropy objective function [37], where N is the number of samples in a batch, y is the ground-truth labelings, and $p(y)$ is the predicted probabilities of each sample being an anomaly:

$$L(y) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1-y_i)\cdot log(1-p(y_i))$$

Under the FL setting, each client's model optimizes its individual $p(y)$ function for its own subset of examples. When aggregating the learned weights, the global model should arrive at a minimal $L(y)$ despite only performing local optimization. Since cross-entropy is being used as a surrogate loss function here and is sensitive to the number of labeled instances in each class, data augmentation is especially important such that anomalies are given proper consideration when fine-tuning $p(y)$.

We now describe our augmentation strategy under the FL environment in detail. In the typical scenario, the client (device) generates its own data. Because supervised learning is assumed to detect anomalies, we also assume the generated data instance is tagged to indicate whether it is normal or not. When the client participates in an FL round, it analyzes the generated instances with respect to class population. In general, the number of anomalies is much smaller than that of normal samples, as can be seen in Table 2. The client then augments the minority class data to mitigate the imbalance problem. Each client's dataset is rebalanced such that there are an equal amount of normal and anomalous readings, using either random oversampling, SMOTE, ADASYN or a pretrained GAN. After balancing the number of instances in both classes, the client creates the local model, which is then aggregated to a global model for that round.

For augmenting the minority class data (i.e., anomalies), random oversampling and SMOTE duplicate or create instances chosen in a uniform manner, while stratified sampling and ADASYN introduce new distributions assigning “importance” values to either clusters of or individual anomalies. SMOTE and ADASYN generate synthetic data using a randomly selected k-nearest neighbor, where we set k to 5 to ensure some variations without using points that are too distant in the feature space. For stratified sampling, k is set to 25 to obtain a better sense of the class density in each anomaly's surrounding neighborhood.

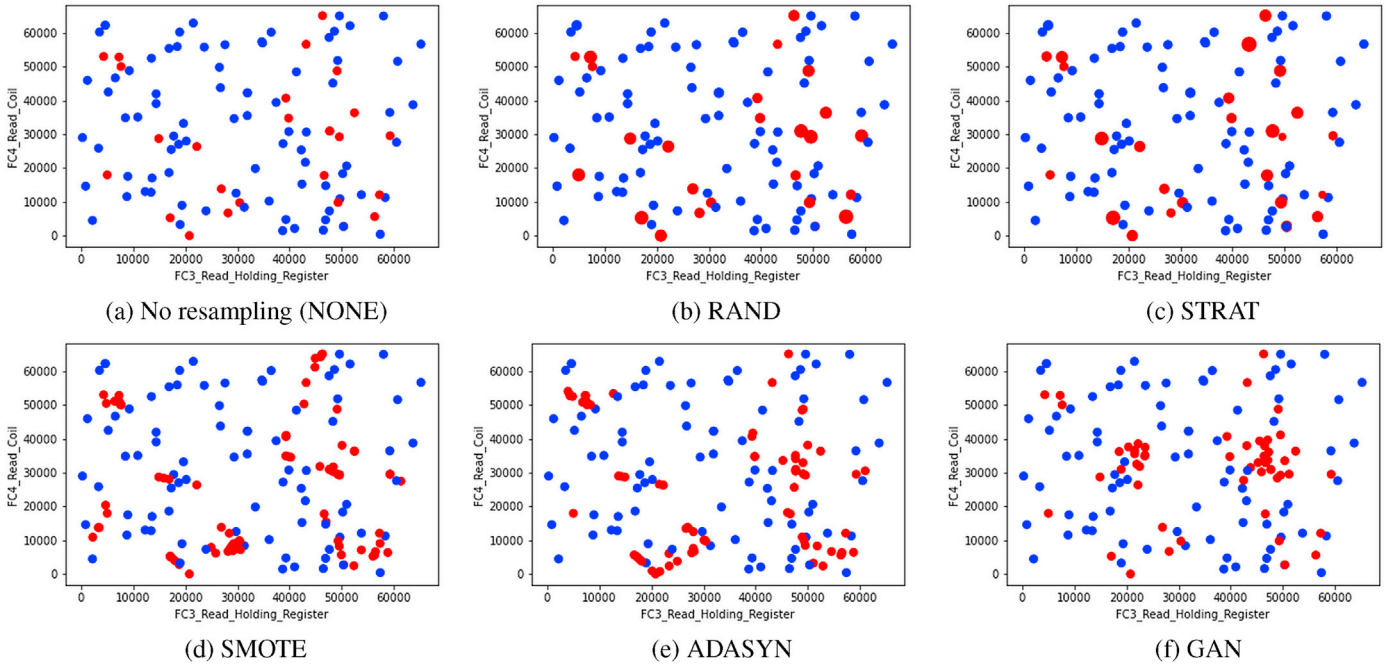


Fig. 3. Various resampling strategies for client 1 receiving 1/50 of the Modbus dataset. Negative samples are shown in blue and positive samples are in red. The FC1 and FC2 features are limited to a 10,000 x 10,000 window. Larger points indicate a greater frequency of sampling.

Table 5
Two-sample KS test statistics and maximum *p*-values (comma separated) after augmentation. The two samples are considered from the same distribution where the test statistic is close to 0, or the *p*-value is close to 1. Typically, a *p*-value less than 0.05 indicates that the two distributions are different.

Method	Modbus	Weather	DS2OS
RAND	0.001, ≈ 1	0.001, ≈ 1	0.001, ≈ 1
STRAT	0.002, ≈ 1	0.005, .05	0.098, ≈ 0
SMOTE	0.013, 7e-8	0.012, 5e-8	0.025, 4e-5
ADASYN	0.014, 7e-8	0.028, ≈ 0	0.104, ≈ 0
GAN	0.107, ≈ 0	0.153, ≈ 0	0.110, ≈ 0

The GAN model consists of a generator and a discriminator with the following parameters. The generator has five hidden layers of dimensions: 128, 256, 512, 526, and 128. The input layer is a 5-dimension random noise vector and the output layer is a vector with the same dimensionality as the desired number of features. The discriminator takes in the same number of features, has hidden layers of sizes 512, 256, and 128, and outputs a single value corresponding to the likelihood that its input is a real anomaly or a synthesized one. All activation functions between layers are Leaky ReLUs with a negative slope coefficient of 0.2, aside from the final layers which use sigmoid activations. Additionally, dropout layers with a probability 0.3 follow each hidden layer in the discriminator model.

Fig. 3 provides a visual example of these data augmentation techniques. The TON_IoT Modbus dataset is partitioned across 50 client nodes and the aforementioned algorithms are used to balance the two classes. A cross section is then taken such that the FC1 and FC2 register values are confined between the range 30,000 and 40,000, and the FC3 and FC4 values are plotted, as shown in the figure. In the cases of random and stratified sampling, larger points indicate a greater frequency of sampling that anomaly.

As can be seen from Fig. 3, performing data augmentation may, but does not necessarily, change the underlying distributions for each feature. The two-sample Kolmogorov-Smirnov (KS) test can be used to quantify the difference between probability distributions using sets of samples. Table 5 shows the results of KS tests for anomalous data, reporting the test statistics and maximum *p*-values across all features for the different datasets and augmentation strategies. As normal readings are unchanged, we omit these samples when conducting the tests. SMOTE, ADASYN, and GAN, as expected, significantly alter the data distributions, synthesizing new points. As RAND chooses samples uniformly, it should not change the data distributions. STRAT sampling depends on the proximity of anomalies to nearby normal readings; hence, the change in the underlying distributions appears to be dataset-dependent. Note that altering the feature distributions is not a requirement for augmentation; rebalancing the two classes is sufficient for adding weight to the anomalies when applying the cross-entropy loss function.

Computational complexity is a critical concern in an FL setting with resource-constrained devices. In fact, the majority of the oversampling methods we employ do not add on any significant overhead. Considering n to be the number of data samples and m to be the data dimensionality, pure random oversampling runs in $O(n)$ time, while stratified sampling, SMOTE, and ADASYN all run in $O(nm)$ time as they rely upon the k-nearest neighbors algorithm before selecting points. GAN training is substantially more rigorous, requiring many passes through the dataset and backpropagation steps that add on a non-negligible layer of complexity. Our experimental result also shows that the time taken to train a GAN model is ~4 orders of magnitude greater than a random sampling (as will be discussed in Section 5.4). As we only perform augmentation once prior to FL training, the additional time expense thereafter is only due to a larger set of anomalies clients train with. We keep track of the average runtimes in our experiments to determine if the greater magnitude of samples imposes noticeable latency.

5. Experiments

We conduct a series of experiments for homogeneous and heterogeneous settings on the NERSC Cori supercomputer system (cori.nersc.gov). The TensorFlow Federated library is used to simulate training FL models across client nodes. For all models, we use the same neural network architecture described in Section 3.3.

In this section, we first report the experimental results observed from homogeneous settings individually for the Modbus, Weather, and DS2OS datasets. The effectiveness of GAN data augmentation is discussed

Table 6
TON_IoT Modbus FL performance metrics for various nodes and dataset rebalancing strategies over 100 rounds.

# Clients (nodes)	Method	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)	Avg Round Time (s)
5	NONE	94.74	86.99	89.94	88.39	**13.58**
	RAND	94.65	86.49	90.24	88.33	23.35
	STRAT	**95.94**	**89.85**	**92.15**	**90.99**	22.86
	SMOTE	91.97	79.68	86.28	82.84	20.56
	ADASYN	87.91	67.07	90.72	77.12	24.77
50	NONE	85.98	76.47	54.36	63.55	**22.77**
	RAND	87.29	77.97	60.92	68.40	25.81
	STRAT	**88.14**	**79.78**	64.69	**71.45**	23.53
	SMOTE	76.30	48.07	**73.42**	58.10	24.80
	ADASYN	76.81	48.77	72.20	58.22	24.98
100	NONE	81.87	69.07	35.20	46.64	**34.68**
	RAND	83.49	70.99	46.17	55.96	37.09
	STRAT	**83.89**	**71.99**	47.60	**57.31**	36.58
	SMOTE	72.22	42.65	**66.88**	52.09	36.87
	ADASYN	73.34	44.20	65.46	52.77	36.63

separately in the following subsection, and then we investigate the heterogeneous setting in which we assume discrepancies among devices with respect to their data generation rates.

5.1. IoT modbus results

For the experiments in this section, the Modbus dataset was partitioned such that each client node received an equal number of samples. Data for each client is selected at random; hence, this partitioning simulates data being Independently and Identically Distributed (IID), which is a common setting for studying FL performance. The heterogeneous setting is considered in Section 5.5, where the IID assumption is relaxed by creating dramatic differences in the amount of data available to each client.

Trials are conducted using 5, 50, and 100 nodes in training, which result in a relatively large, medium, and a small amount of data to train each client with, respectively. For each node size, clients will iteratively rebalance their own dataset using either random oversampling, stratified sampling, SMOTE, or ADASYN. A benchmark case with no resampling is also considered. Metrics are recorded for each pair of node size and oversampling method, including accuracy, precision, recall, F1 score, and average training time per round (in seconds).

The collected metrics over a testing dataset are reported in Table 6, where boldface indicates the best value achieved for a fixed number of nodes. The NONE rows in the table represent naive FL classifiers. A general trend evident from this data is that as training becomes more decentralized over more nodes, oversampling strategies become increasingly effective at recognizing anomalies. For only 5 nodes, the difference between the baseline and the best performing method after 100 rounds is fairly narrow with only a 2.60% gap in F1 score. When 100 nodes are used in training, this gap increases sharply to 10.67%. Especially for 50 and 100 nodes, random oversampling and its weighted stratified variant appear to be the best means of dataset rebalancing. This suggests that the anomalies in the original data are sufficient for learning models, as evidenced by the centralized case that does not use oversampling to achieve high metrics. The naive model with no augmentation takes longer to reach the same performance level as random oversampling, and alternatively, randomly undersampling the normal readings led to a classifier that stagnated at ~50% testing accuracy. This indicates that the large volume of normal readings is necessary to avoid underfitting; however, the issue of too few anomalies can be remedied by a simple random oversampling.

SMOTE and ADASYN did not appear to provide significant improvements for classification. Notably, Table 6 shows that the SMOTE-trained classifier over 100 nodes had the highest recall but the lowest precision after 100 rounds, with the ADASYN classifier showing very similar metrics. This indicates that synthetic data generation for the Modbus dataset will allow more anomalies to be detected at the expense of a higher degree of false positives. Due to anomalies being sparsely distributed rather than clustered together, these algorithms seemed to have shifted the classification boundary too significantly such that more normal readings became misclassified.

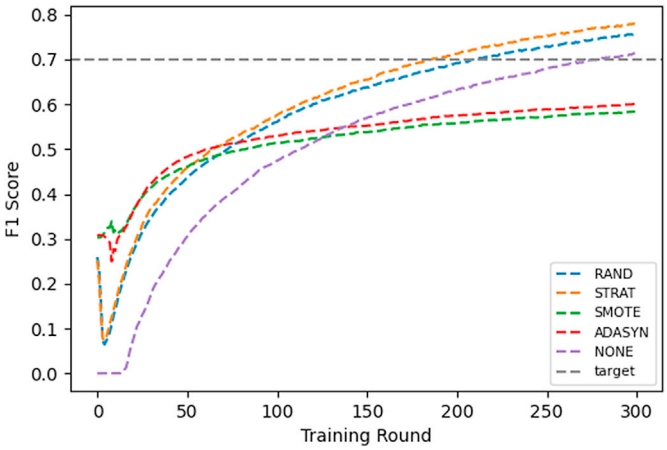


Fig. 4. F1 scores using various preprocessing methods over 100 client nodes for the TON_IoT Modbus dataset. Random/stratified oversampling and a naive approach surpass a 70.00% target threshold while SMOTE and ADASYN do not.

Fig. 4 shows F1 scores over an extended period (300 rounds) of training to compare to a target threshold of 70.00% using 100 nodes. Our naive FL classifier attained this F1 score after 280 rounds, while it took 211 rounds for an FL classifier using random oversampling of client data to surpass the threshold and only 187 with stratified sampling. Neither SMOTE nor ADASYN reached this performance level after 300 rounds of training and appeared to be converging at a lower value. Stratified oversampling was able to yield a 33.2% decrease in the number of rounds to attain this target with only a 5.5% increase in average training time per round. Note that wall-clock time is not a particularly relevant metric, as sending updates back to the server for aggregation is a far more significant bottleneck than on-device training.

5.2. IoT weather results

Next, we run similar experiments on the TON_IoT Weather dataset. We assume that most real-world scenarios will run FL in a highly decentralized manner and restrict our attention to the case of homogeneously partitioning data across 100 nodes. As can be observed in Table 2, the Weather dataset has more than double the number of total instances compared to Modbus, although the percentage of anomalies is nearly half. A greater magnitude of overall data is certainly advantageous when training a classifier, though the range of possible values each

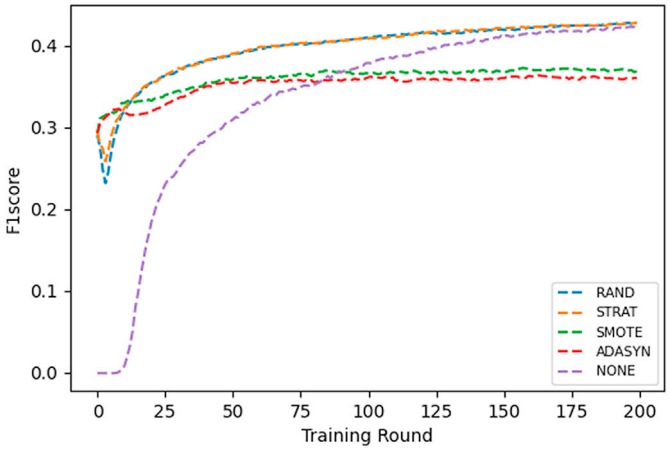


Fig. 5. F1 scores using various preprocessing methods over 100 client nodes for the TON_IoT Weather dataset.

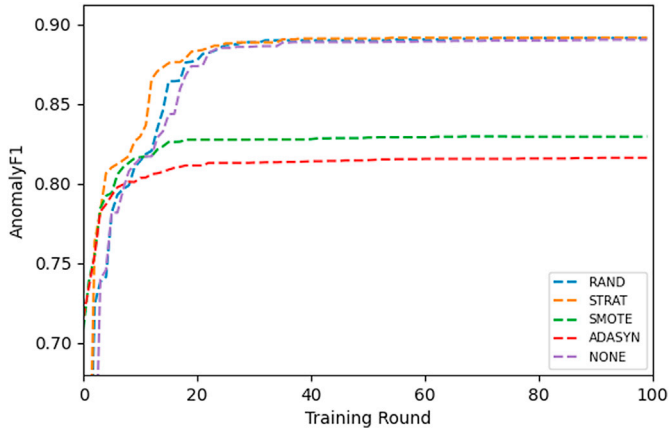


Fig. 6. F1 scores using various preprocessing methods over 100 client nodes for the DS2OS traffic trace dataset.

feature can take is quite limited. As such, we first observe the centralized case to determine a reasonable maximum performance we could hope to achieve under FL.

Indeed, the lower proportion of anomalies appears to be a major limitation for a centralized classifier. A model with identical architecture to the one used in the FL setting was trained for 100 epochs over the encoded Weather readings, and the F1 score ultimately converged to 0 because the classifier learned to predict all inputs to be normal. FL could be a more advantageous method of learning for such a dataset, as first partitioning the instances to clients and making smaller incremental updates can lead to a more useful model. We would anticipate varying parameters such as using a smaller batch size or learning rate to help optimize the centralized classifier; however, we merely opt to show that this is a difficult learning task and focus attention on the FL setting, as well as emphasize that identifying a significant portion of the anomalies does not appear to be feasible.

Next, FL models are trained over 100 nodes using the same oversampling strategies as the Modbus dataset. In the previous case, a larger proportion of anomalies allowed for fairly reliable classification; however, the Weather dataset lacks diversity features with a wide range of values. Thus, we do not set a target threshold for this experiment and merely observe where and when training begins to converge. A graph of F1 scores over 300 rounds for the baseline, random oversampling, stratified sampling, SMOTE, and ADASYN methods are shown in Fig. 5.

Similar to the centralized case, the naive classifier initially ignores all anomalies resulting in a 0 F1 score. It takes a non-negligible number of rounds for this model to reach a performance level comparable to other models, although it does eventually surpass the models trained using the nearest neighbor techniques as was in the case with the Modbus data. A notable difference here is that stratified sampling does not seem any better than random sampling; while the probability distributions of the anomalies in the two methods do appear to be statistically different, sampling boundary points more frequently may not always be advantageous. Stratified sampling is likely to be much more powerful when the original classes exhibit greater clustering patterns than is present in the TON_IoT datasets.

5.3. DS2OS results

We begin our experiments for the DS2OS dataset by confirming pre-existing results, particularly the greater separability of the two classes. Using the pruned and encoded set of features as described in Section 3.1, we trained a centralized classifier with the same setup as described in the previous subsection and observed a maximum binary accuracy of 98.9% and F1 score of 75.6%. The vast majority of normal instances were classified correctly, with a false negative (normal instance) rate under 1%; however, ~9.5% of anomalies were misclassified.

As anomalies only make up less than 3% of this dataset, it would be reasonable to assume that a classifier attempting to minimize overall loss could ignore anomalies and achieve ~97% binary accuracy on its training set. However, the more substantial disparities between normal and anomalous classes make this a less challenging task than for the TON_IoT Weather dataset, and in the centralized and FL settings, quite high AD metrics can be achieved.

Fig. 6 reports F1 scores for each oversampling technique using FL for 100 nodes. Due to the aforementioned class differences, learning converges quite quickly, and it does not appear beneficial to train beyond 100 global epochs, and therefore, no target threshold is set. One major difference used in these trials was a reduced level of oversampling; rather than perfectly balancing the two classes, better performance was achieved by only sampling until a 90% normal/10% anomaly split was reached. An equal ratio between the classes resulted in the random oversampling method yielding a lower F1 score than the baseline, which is likely because the wide range of normal instances available is enough to recognize erroneous samples quite well without much knowledge about the anomalies. Then, it would be more beneficial to learn what types of inputs should be considered normal rather than equally focusing on regular readings and outliers when the sheer quantity of the former set cause it to possess richer information.

Nonetheless, careful random or stratified oversampling can cause the FL-trained classifier to converge quicker — in 25 rounds using stratified sampling as opposed to 36 in the baseline. However, they do not appear well-suited for datasets such as this one. Instead, semi-supervised learning has greater potential to improve performance as one class has enough information to make assumptions about the other, which we investigate in the next subsection and include in our directions for future work.

5.4. GANs for data augmentation

While GANs can certainly be effective at data augmentation, they appear to rely more heavily than the other techniques on the assumption that the class it learns to augment (i.e., the anomalies) is distinct from others present in the dataset. Random and stratified oversampling techniques do not add new unique instances, while SMOTE and ADASYN constrain their synthesized examples to a local neighborhood around existing points. Stratified sampling and ADASYN also make use of the majority class to influence selection probability, while a GAN trained for data augmentation has no such information beyond a client's anomalies. This can induce a greater risk of class overlap; moreover, the limited training data available in highly decentralized settings could result in individual GANs recognizing and learning patterns that do not generalize well to other clients' data.

Table 7
Average F1 scores for naive FL and GAN augmentation.

Dataset	Benchmark (%)	GAN (%)
TON_IoT Modbus	71.3	66.3 (−5.0)
TON_IoT Weather	44.0	40.9 (−3.1)
DS2OS Traffic Traces	89.0	89.5 (+0.5)

The two TON_IoT datasets used in this study appear to suffer from these difficulties associated with using pretrained GANs on each node for data augmentation. With sparsely distributed anomalies amongst the normal readings and a very limited quantity of the former class instances, training effective GANs that adhere to the true anomalous data distribution seems to be a challenging task. The approach fairs somewhat better for the DS2OS dataset, where the anomalies behave more typically like outliers, slightly outperforming the rest of the preprocessing methods. Average F1 scores compared to the naive FL models for each dataset after 300 rounds (or 100 rounds for the DS2OS data, after convergence) are reported in Table 7 using GANs trained for 50 epochs.

Another issue posed by GANs is the computational expense of training them before they can be useful for augmentation. While clients can train their GANs in parallel, devices with a larger set of anomalies could add an additional bottleneck that further slows down the training process. In a moderately decentralized setting where each client received 1/50 of the Modbus dataset, the time to fully train each GAN is approximately 4 orders of magnitude greater, in seconds, than a simple random sampling. Accompanied by the fact that the majority of clients likely would not possess enough data to sufficiently train a GAN, we do not consider this to be a very useful method for AD when anomalies are sparse and infrequent.

5.5. Heterogeneous setting

In real-world scenarios, it is very unlikely for data to be evenly distributed across client devices participating in training. The *FedAvg* algorithm partially considers this, computing a weighted average between updates based on the proportion of data a client uses in training relative to the total data used by all the clients. Nonetheless, heterogeneous partitioning can pose an issue for some classifiers to attain acceptable performance.

We iteratively generate Gaussian random variables and translate them to their corresponding probabilities, which represent the proportion of the total available data to assign to a new client. We always consider the smaller-tail probability such that the dataset is not exhausted too quickly. Using the Modbus dataset, we are able to create partitions for 35 nodes consistently such that each client has multiple anomalies to perform generative oversampling techniques.

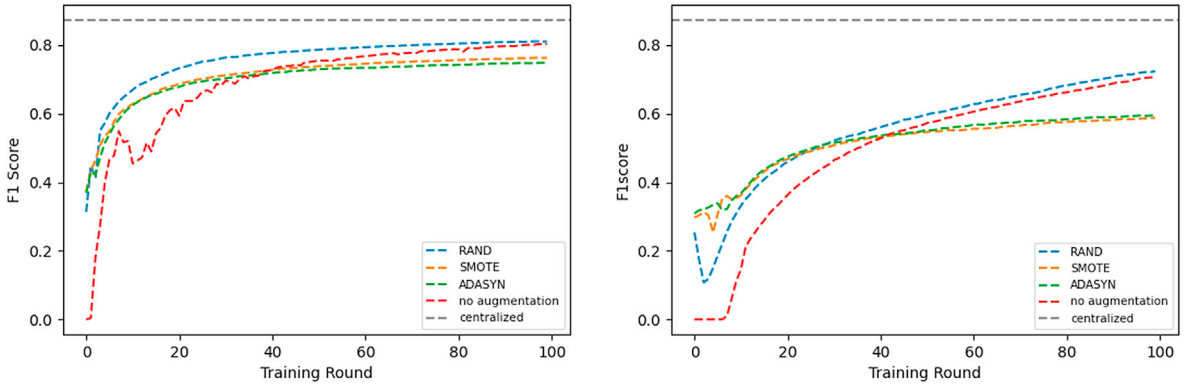
Fig. 7 shows F1 score results for an FL classifier trained over the Modbus data where the number of instances on each client is fixed and determined by a Gaussian distribution. Furthermore, we also show F1 scores for the homogeneous case using FL over 35 nodes for comparison. Interestingly, heterogeneous partitioning actually yields higher testing metrics over 100 rounds than the homogeneous case regardless of oversampling algorithm. This appears to be due to the fact that the highest weighted updates are ones that come from nodes with the most amount of data, so classifiers can benefit from a heterogeneous setting where some clients have large numbers of examples that are indicative of the overall data distributions of each class.

While not entirely following the IID assumption, FL nonetheless appears well-equipped to handle clients with varying sizes of their datasets. While random oversampling does cause a speedup of reaching an F1 score of 70% from 34 rounds to only 15 rounds — a 55.9% decrease — this can be misleading as all models trained over heterogeneous data ultimately converge at approximately the same level of performance. Indeed, there is only a 0.98% difference in the F1 score between the random oversampling model and the naive model after 100 rounds of training. In environments where the number of training rounds must be rigorously optimized, random oversampling can certainly be advantageous; however, naive FL over heterogeneous data does not appear to suffer from the same degree of performance degradation as in the homogeneous case.

Performing such oversampling would be more beneficial for clients with data crucial for generalizing beyond training sets but have too few total samples to contribute a significant update after aggregation, i.e., their data follows a different distribution than data belonging to other clients. Future work using other datasets that are generated from multiple sources, for instance, could test such an approach to determine how strong the benefit of data augmentation can be in this more extreme non-IID setting.

6. Conclusion and future directions

We have shown that naive FL may initially stagnate when training over a large number of clients with imbalanced datasets, which can ultimately degrade overall model performance due to poorly learned initial parameters. Random oversampling, SMOTE, and ADASYN can improve the AD performance in earlier rounds in homogeneous and heterogeneous settings, where we observed up to a 22.9% improvement in F1 score. Interestingly, stratified sampling performed at least comparable to baseline models consistently. This appears to be due to the sparse distribution of anomalies among normal readings in the IoT Modbus dataset, which is likely a more realistic setting than anomalous data being highly clustered together. With the popular use of GANs, we also examined their



(a) Heterogeneous partitioning resulting in variation in client dataset sizes

(b) Homogeneous partitioning resulting in equal client dataset sizes

Fig. 7. F1 scores over the TON_IoT Modbus dataset for 100 rounds of training FL classifiers across 35 nodes, where the data is first heterogeneously and then homogeneously partitioned. Heterogeneous splitting results in metrics closer to those of the centralized classifier.

effectiveness for data augmentation in our setting. The GAN-based approach has the potential to improve detection performance with rigorous optimization and under idealized conditions; however, we observed quite expensive computational overheads (~4 orders of magnitude greater than random oversampling), implying limited benefits in FL settings.

From the analysis of the DS2OS data, we observed that anomalies might often show significant deviations from normal instances. In that regard, it is interesting to investigate how to improve AD performance using semi-supervised recognition of anomalies over FL. This has the major advantage of learning only the behavior of the normal data, allowing for potentially high anomaly recognition even with very few of these samples in the training set. Supervised learning lends itself more naturally to cases when there is sufficient data in each class to make generalizations about them, which is not often the case with anomalies. Moreover, it is plausible that some clients could have no anomalous samples, which would deter them from contributing any valuable information to our proposed approach. In that sense, training networks that make no assumptions about anomalies besides them being outliers, such as autoencoders, would be an option.

Declaration of interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported by the Office of Advanced Scientific Computing Research, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231, and also used resources of the National Energy Research Scientific Computing Center (NERSC).

References

[1] A.P. Plageras, K.E. Psannis, C. Stergiou, H. Wang, B.B. Gupta, Efficient iot-based sensor big data collection–processing and analysis in smart buildings, Future Generat. Comput. Syst. 82 (2018) 349–357.
[2] E. Bautista, M. Romanus, T. Davis, C. Whitney, T. Kubaska, Collecting, monitoring, and analyzing facility and systems data at the national energy research scientific computing center, in: Proceedings of the 48th International Conference on Parallel Processing, Workshops, 2019, pp. 1–9.
[3] E. Luo, M.Z.A. Bhuiyan, G. Wang, M.A. Rahman, J. Wu, M. Atiquzzaman, Privacyprotector: privacy-protected patient data collection in iot-based healthcare systems, IEEE Commun. Mag. 56 (2) (2018) 163–168.
[4] M. Elhoseny, G. Ramírez-González, O.M. Abu-Elnasr, S.A. Shawkat, N. Arunkumar, A. Farouk, Secure medical data transmission model for iot-based healthcare systems, IEEE Access 6 (2018) 20596–20608.
[5] A. Mylonas, O.B. Kazanci, R.K. Andersen, B.W. Olesen, Capabilities and limitations of wireless co2, temperature and relative humidity sensors, Build. Environ. 154 (2019) 362–374.
[6] A. Hard, K. Rao, R. Mathews, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, D. Ramage, Federated Learning for Mobile Keyboard Prediction, arXiv preprint arXiv: 1811.03604.
[7] K. Dolui, I.C. Gyllensten, D. Lowet, S. Michiels, H. Hallez, D. Hughes, Poster: towards privacy-preserving mobile applications with federated learning–the case of matrix factorization, in: The 17th Annual International Conference on Mobile Systems, Applications, and Services, 2019.
[8] S. Ramaswamy, R. Mathews, K. Rao, F. Beaufays, Federated Learning for Emoji Prediction in a Mobile Keyboard, arXiv preprint arXiv:1906.04329.
[9] S. Zheng, N. Apthorpe, M. Chetty, N. Feamster, User perceptions of smart home iot privacy, Proceed. ACM Hum. Comput. Interact. 2 (2018) 1–20. CSCW.
[10] H. Lin, N.W. Bergmann, Iot privacy and security challenges for smart home environments, Information 7 (3) (2016) 44.
[11] B. McMahan, E. Moore, D. Ramage, S. Hampson, B.A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in: Artificial Intelligence and Statistics, 2017, pp. 1273–1282.
[12] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečný, S. Mazzocchi, H. B. McMahan, et al., Towards Federated Learning at Scale: System Design, arXiv preprint arXiv:1902.01046.
[13] T. Li, A.K. Sahu, A. Talwalkar, V. Smith, Federated learning: challenges, methods, and future directions, IEEE Signal Process. Mag. 37 (3) (2020) 50–60.
[14] H. Aoyama, A study of stratified random sampling, Ann. Inst. Stat. Math. 6 (1954) 1–36.
[15] N.V. Chawla, K.W. Bowyer, L.O. Hall, W.P. Kegelmeyer, Smote: synthetic minority over-sampling technique, J. Artif. Intell. Res. 16 (2002) 321–357.
[16] H. He, Y. Bai, E.A. Garcia, S. Li, Adasyn: Adaptive synthetic sampling approach for imbalanced learning, in: 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), IEEE, 2008, pp. 1322–1328.
[17] V. Sandfort, K. Yan, P.J. Pickhardt, R.M. Summers, Data augmentation using generative adversarial networks (cyclegan) to improve generalizability in ct segmentation tasks, Sci. Rep. 9 (1) (2019) 1–9.
[18] F.H. K.d.S. Tanaka, C. Aranha, Data augmentation using gans, Proc. Mach. Learn. Res. (2019) 1–16.
[19] The TON_IoT datasets. https://research.unsw.edu.au/projects/toniot-datasets. (Accessed 19 January 2022).
[20] IoT traffic traces gathered in a the DS2OS IoT environment. https://www.kaggle.com/francoisxa/ds2ostraffictraces/. (Accessed 19 January 2022).
[21] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection: a survey, ACM Comput. Surv. 41 (3) (2009) 1–58.
[22] R. Chalapathy, S. Chawla, Deep Learning for Anomaly Detection: A Survey, arXiv preprint arXiv:1901.03407.
[23] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, Y. Elovici, N-baiot—network-based detection of iot botnet attacks using deep autoencoders, IEEE Pervasive Comput. 17 (3) (2018) 12–22.
[24] L. Lyu, J. Jin, S. Rajasegarar, X. He, M. Palaniswami, Fog-empowered anomaly detection in iot using hyperellipsoidal clustering, IEEE Internet Things J. 4 (5) (2017) 1174–1184.
[25] M. Hasan, M.M. Islam, M.I.I. Zarif, M. Hashem, Attack and anomaly detection in iot sensors in iot sites using machine learning approaches, Internet Things 7 (2019) 100059.
[26] R. Doshi, N. Apthorpe, N. Feamster, Machine learning ddos detection for consumer internet of things devices, in: 2018 IEEE Security and Privacy Workshops (SPW), IEEE, 2018, pp. 29–35.
[27] D. Preuveneers, V. Rimmer, I. Tsingenopoulos, J. Spooren, W. Joosen, E. Ilie-Zudor, Chained anomaly detection models for federated learning: an intrusion detection case study, Appl. Sci. 8 (12) (2018) 2663.
[28] T.D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, A.-R. Sadeghi, Dïot: a federated self-learning anomaly detection system for iot, in: 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), IEEE, 2019, pp. 756–767.
[29] Y. Zhao, J. Chen, D. Wu, J. Teng, S. Yu, Multi-task network anomaly detection using federated learning, in: Proceedings of the Tenth International Symposium on Information and Communication Technology, 2019, pp. 273–279.
[30] S. Li, Y. Cheng, W. Wang, Y. Liu, T. Chen, Learning to Detect Malicious Clients for Robust Federated Learning, arXiv preprint arXiv:2002.00211.
[31] M.-O. Pahl, F.-X. Aubet, All Eyes on You: Distributed Multi-Dimensional Iot Microservice Anomaly Detection, 2018, pp. 72–80.
[32] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial networks, Commun. ACM 63 (11) (2020) 139–144.
[33] J.L.P. Lima, D. Macêdo, C. Zanchettin, Heartbeat anomaly detection using adversarial oversampling, in: 2019 International Joint Conference on Neural Networks, IJCNN), 2019, pp. 1–7.
[34] A. Ghazikhani, H.S. Yazdi, R. Monsefi, Class imbalance handling using wrapper-based random oversampling, in: 20th Iranian Conference on Electrical Engineering (ICEE2012), 2012, pp. 611–616.
[35] J. Kong, W. Kowalczyk, S. Menzel, T. Bäck, Improving Imbalanced Classification by Anomaly Detection, 2020, pp. 512–523.
[36] M. Salem, S. Taheri, J.S. Yuan, Anomaly generation using generative adversarial networks in host-based intrusion detection, in: 2018 9th IEEE Annual Ubiquitous Computing, Electronics Mobile Communication Conference, UEMCON), 2018, pp. 683–687.
[37] G. Pang, C. Shen, L. Cao, A.V.D. Hengel, Deep learning for anomaly detection: a review, ACM Comput. Surv. 54 (2) (2021) 1–38.