Enhancing IoT Anomaly Detection Performance for Federated Learning

Brett Weinger
Stony Brook University
Stony Brook, NY
brett.weinger@stonybrook.edu

Jinoh Kim
Texas A&M University
Commerce, TX
jinoh.kim@tamuc.edu

Alex Sim
Lawrence Berkeley Nat'l Laboratory
Berkeley, CA
asim@lbl.gov

Makiya NakaShima
Texas A&M University
Commerce, TX
makiya.nakashima@gmail.com

Nour Moustafa
University of New South Wales
Sydney, Australia
nour.moustafa@adfa.edu.au

K. John Wu
Lawrence Berkeley Nat'l Laboratory
Berkeley, CA
kwu@lbl.gov

***Abstract*—While federated learning (FL) has gained great attention for mobile and Internet of Things (IoT) computing with the benefits of scalable cooperative learning and privacy protection capabilities, there still exist a great deal of technical challenges to make it practically deployable. For instance, the distribution of the training process to a myriad of devices limits the classification performance of machine learning (ML) algorithms, often showing a significantly degraded accuracy compared to centralized learning. In this paper, we investigate the problem of performance limitation under FL and present the benefit of *data augmentation* with an application of anomaly detection using an IoT dataset. Our initial study reveals that one of the critical reasons for the performance degradation is that each device sees only a small fraction of data (that it generates), which limits the efficacy of the local ML model (constructed by the device). This becomes more critical if the data holds the *class imbalance* problem, observed not infrequently in practice (e.g., a small fraction of anomalies). Moreover, *device heterogeneity* with respect to data quantity is an open challenge in FL. Based on these observations, we examine the impact of data augmentation on detection performance in FL settings (both homogeneous and heterogeneous). Our experimental results show that even a simple random oversampling can improve detection performance with manageable learning complexity.**

***Index Terms*—Federated Learning, Internet of Things, Anomaly Detection, Machine Learning**

I. Introduction

Data collection and measurement are essential services in the Internet of Things (IoT) computing sector for building operations [1], [2], healthcare [3], [4], environmental studies [5], and mobile behavior analysis [6], [7], to list a few. Collected instances are then analyzed often using one or more machine learning (ML) algorithms for high-level data analytics. However, the ever-increasing concern of privacy and data confidentiality may be a big obstacle to implement such a data-driven analytics process that assumes the sharing of raw data. For instance, the data measured by healthcare devices needs to be protected by law, while other applications such as smart homes and public safety also require a degree of privacy and security protection [8], [9].

Federated learning (FL) has been introduced to enable privacy-preserving learning without sharing private data with external entities [10]–[12]. In other words, rather than sharing potentially privacy-sensitive raw data, devices process their local data to create an ML model ("local model"), which is then collectively combined to build an aggregated model ("global model") often by a coordinator (e.g., a cloud server). The privacy concern can thus be mitigated by limiting the sharing of raw data in the FL setting. In the meantime, numerous challenges have also been introduced, which should be addressed for practical deployment of FL [12]. For instance, the distribution of the training process to a myriad of devices limits the classification performance of machine learning (ML) algorithms, often showing a significantly low accuracy compared to the centralized learning. In this paper, we investigate the problem of performance limitation under FL and present the benefit of data augmentation with an application of anomaly detection (AD) using an IoT dataset.

We first show that one of the critical reasons for the performance degradation is that each device sees only a small fraction of data (that it generates), which limits the efficacy of the local ML model (constructed by the device). This becomes more critical if the data holds the *class imbalance* problem, observed not infrequently in practice (e.g., low frequency of anomalies while most instances generated are normal). Moreover, *device heterogeneity* with respect to data quantity is an open challenge in FL. Based on our initial observations, we examine the impact of *data augmentation* on detection performance in FL settings (both homogeneous and heterogeneous), with a set of augmentation techniques including random oversampling, SMOTE [13], and a variant of SMOTE known as ADASYN [14].

Our experimental results show that even a simple random oversampling significantly improves detection performance with manageable complexity when training over a large number of client nodes. Under the assumption of the equal-rate data generation by clients (i.e., homogeneous), random sampling yields up to a 19.98% improvement in F1 score compared to the baseline without augmentation after 100 rounds of training. In case of the heterogeneous setting which assumes different data generation rates among clients, random sampling helps FL converge quickly within significantly fewer rounds (more than half) than the baseline.

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

- We show the performance degradation problem in case of a simple adoption of FL by comparing AD rates to the traditional, non-FL approach with the IoT dataset (TON_IoT) recently collected in 2019;
- We present our approach to improve the classification performance by employing the concept of data augmentation. The augmentation techniques include random sampling, SMOTE [13], and ADASYN [14];
- We show that data augmentation can be used to achieve high performance results faster over highly decentralized FL training through extensive experiments in both homogeneous and heterogeneous settings.

The organization of this paper is as follows. In Section II, we introduce relevant background information regarding anomaly detection, federated learning, and the dataset we use for this work. We conduct preliminary analysis including data statistics and centralized classifier metrics in Section III, and then present common strategies for handling imbalanced classification and explain our sampling approach in Section IV. In Section V, we report our experimental results with these strategies for both homogeneous and heterogeneous settings. Section VI summarizes the previous studies closely related to our work, and we conclude our presentation with a summary and future direction in Section VII.

II. Background

In this section, we provide an overview of anomaly detection, federated learning, and the IoT dataset used in this study.

A. Anomaly Detection

AD is a widely important and actively studied area of computational research [15]. With the significant advance on ML (including deep learning), a body of studies investigated the adoption of ML to improve the AD performance [16]. Basically, the vast majority of AD algorithms rely on the infrequency of anomalies occurring as well as anomalous data deviating substantially from normal instances. When these assumptions hold true, unsupervised learning is often applied to identify clusters of anomalies without any class labeling. In practice, anomalies can make up a significant portion of a dataset (i.e. a client that has been corrupted by a malicious attack) and anomalous feature values may be relatively similar to normal samples. Supervised classifiers are typically more effective in this case, which we primarily focus on in this work in a distributed setting.

Performance metrics are used to measure the ability of a learning algorithm to correctly identify anomalies. Accuracy can be a biased metric in this regard, as it gives equal weight to positive and negative classes in the subset of data it is measured over. For convention, we consider anomalies to be the positive class and normal samples to be the negative class. Precision and recall are instead more appropriate for this task, which measure the proportion of anomaly classifications that are correct and proportion of total anomalies identified, respectively. F1 score is the harmonic mean of precision and recall, where a higher value indicates more reliable and frequent AD.

TABLE I
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	$\frac{2TP}{2TP+FP+FN}$

Improving F1 score beyond a naive implementation of AD classifiers is the central goal of this work. The performance metrics are defined based on confusion matrix that contains TP (*true positive* – anomalies correctly classified), TN (*true negative* – benign instances correctly classified), FP (*false positive* – benign instances incorrectly classified as anomalies), and FN (*false negative* – anomalies incorrectly classified as benign). Table I provides the definition of the metrics.

B. Federated Learning

FL is an approach to machine learning recently pioneered by Google that makes use of a client-server architecture to allow eligible devices to contribute to global model updates [10]–[12]. McMahan et al. [10] showed that sending a commonly initialized artificial neural network (ANN) to client nodes over subsequent training rounds and aggregating locally computed updates often leads to model convergence at a mutually optimal minimum. Additionally, due to upload speeds posing a more significant bottleneck than training on each client, McMahan et al. proposed the *FederatedAveraging* algorithm [10], which is a variation of Stochastic Gradient Descent (SGD) that iterates multiple local updates per device prior to aggregation.

A phased approach is common when implementing FL for end user devices. The FL server, which is often a cloud-based distributed service, 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 have 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 period. However, a minimum

number of successful updates are necessary to commit to the global model, which can leave the server waiting for the slowest client to continue the learning process. Consequently, FL can be very slow to converge to comparable performance levels of centralized models. It is an open area of research of reducing the amount of rounds necessary to train models as well as improving naive FL algorithms.

Through this process, FL provides the framework for distributed learning without sharing local data, thus mitigating the privacy concern that becomes more crucial in real settings. Despite the benefits, our observation shows that the FL process may not be able to produce a high-quality ML model, and we investigate the impact of class imbalance and data heterogeneity in this study.

C. Description of IoT Dataset (TON_IoT)

The TON_IoT datasets[1], which are a collection of IoT telemetry readings, operating system logs, and network traffic information, have been published in 2019 to help train and test cybersecurity AI applications. Normal and attack scenarios were executed using an architectural testbed to generate a heterogeneous set of sensor readings. These attacks include some of the most common forms of hacking events, including but not limited to scanning attacks, Denial of Service (DoS) attacks, and backdoor attacks. To the best of our knowledge, there do not appear to be existing works using these datasets for FL anomaly recognition algorithms.

Table II provides a summary of the IoT/IIoT datasets included in the TON_IoT package. There exist seven independent datasets collected from different classes of 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 (anomaly) or 0 (normal), while 'type' shows the attack category if applicable. After analyzing individual datasets, we decide to focus on analyzing the Modbus dataset since it contains a rich set of features compared to data for other IoT devices. Modbus is a data communication protocol and connects a plant supervisory computing machine to a remote terminal unit in electrical power systems (i.e., SCADA or Supervisory Control and Data Acquisition systems).

III. Data Analysis and Learning Models

In this section, we analyze the Modbus IoT data employed in this study for AD under FL and discuss potential limitations of the naive adoption of FL with our initial experimental results.

A. Analysis of IoT Modbus Data

As mentioned, the IoT Modbus dataset is used for experiments in this work, as it has the highest variation in features relative to the other IoT device readings. However, aside from time and date, the Modbus dataset only contains four of these numeric features, each corresponding to a different sensor reading from registers. Thus, there are simply not enough numeric features in a continuous number domain.

[1]https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-ton-iot-Datasets/

TABLE II
SUMMARY OF TON_IOT DATASETS

Dataset	Specific features	# instances	% anomaly
Fridge	fridge_temperature (Number) temp_condition (String)	587,077	14.7%
GarageDoor	door_state (Boolean) sphone_signal (Boolean)	591,447	12.9%
GPSTracker	latitude (Number) longitude (Number)	595,687	13.7%
Modbus	FC1_Input_Reg (Number) FC2_Discrete_Val (Number) FC3_Holding_Reg (Number) FC4_Coil (Number)	287,195	22.4%
MotionLight	motion_stat (Number) light_stat (Boolean)	452,263	14.1%
Thermostat	curr_temperature (Number) termostat_stat (Boolean)	442,229	12.7%
Weather	temperature (Number) pressure (Number) humidity (Number)	650,243	13.9%

TABLE III
TRANSFORMING MODBUS FEATURES USING MULTIPLE BINS

Bin size	# features	Training seconds
500	632	599.4
1,000	314	308.1
2,000	154	260.8
3,000	101	190.4
4,000	79	175.6
No encoding	4	93.2

We considered one-hot encoding for these features, but each could take on more than 50,000 different values which is unnecessarily large to train with. For this reason, we transform the features into a discrete domain using a set of bins with a definition of *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 III compares the number of features after the transformation based on *bin_size*, along with the training overhead in seconds. Preliminary testing in Table III shows that setting *bin_size* to 1,000 achieves a substantially large feature space and is more efficient than smaller sizes with the manageable training overhead. This bin size is adopted during experimentation, and further work is needed to study if varying *bin_size* to either increase or decrease the input feature dimensionality can be beneficial.

Prior to conducting our experiments, we analyze statistics for the Modbus dataset to determine which learning algorithms would be most effective. To do so, samples were separated into two groups based on their label such that analytics could be compared based on class. Ideally, a trend in one group that is not present in the other could be leveraged when performing classification, but there do not seem to be extremely notable distinctions between normal samples and anomalies. Figure 1 shows kernel density estimation functions for each class indicating the frequency of values for the FC1_Input_Register feature. The similarity between the two curves demonstrates that anomaly readings do not deviate substantially from their normal counterparts, but this is the desired setting for this work as it motivates well-performing FL algorithms to distinguish

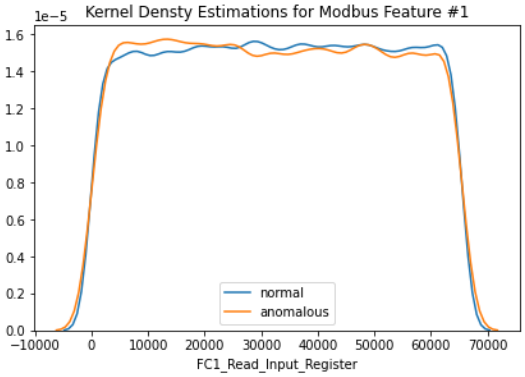


Fig. 1. Kernel Density Estimations for FC1_Input_Register readings, separated by class. The similarity of these probability distributions motivate supervised learning to classify unknown readings.

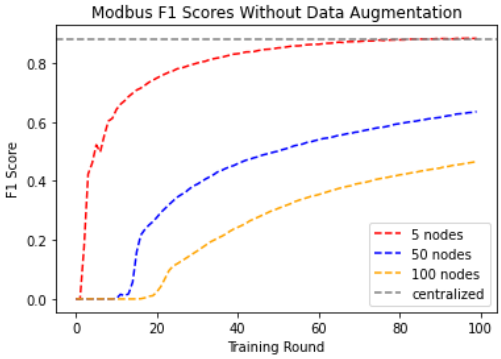


Fig. 2. F1 scores for naive FL classifiers trained over 5, 50, and 100 nodes. An important observation is the initial start at an F1 score of zero due to the class imbalance issue, which could be mitigated by a manual rebalancing on each client.

between classes.

B. Limitation of Naive Federated Learning

The Modbus dataset, as well as the other TON_IoT datasets, all have significantly more normal samples than anomalous readings. This can make learning a supervised classifier a difficult task when a reasonably low loss can be achieved by a SGD optimizer simply by ignoring the positive class. Doing so will yield an F1 score of zero, which is contrary the intended AD goal.

For our initial experiments, we design a deep neural network (DNN) performing AD based on supervised learning. The AD classifier consists of 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. Note that we do not intensively optimize the classifier through rigorous hyperparameter tuning as our interest here is to see how well the AD model works under various FL settings.

To compare the AD performance between the centralized vs. 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 the equal amount of data instances, while each client has a different number of instances in the heterogeneous setting.

We first observe metrics reached by a centralized classifier over the entire Modbus dataset. It takes this classifier only a few epochs to begin to correctly recognize anomalies, converging after approximately 50 epochs to a model with maximum testing accuracy of 94.35% and F1 score of 87.35%. This indicates that supervised learning can be a useful means of AD where unsupervised methods are likely to fail, and it additionally establishes goals for FL model metrics.

We next assume the homogeneous FL setting. In this scenario, each individual update to the global model is not as powerful as in the centralized case. We use the same model architecture as previously described while also setting the number of local epochs at each client to 10. Figure 2 shows testing F1 scores after homogeneously splitting data across 5, 50, and 100 nodes and performing FL for each setting. It is notable that, for imbalanced datasets, distributing the same amount of data to more nodes results in a larger number of rounds until classifiers begin to learn meaningful distinctions between the two classes (that is, a larger number of rounds until the F1 score becomes nonzero). Furthermore, the slope of these curves representing progress in model training is also subject to optimization. When the number of clients with small datasets becomes increasingly high, class imbalance may pose an even more significant issue than in the use case we discuss here and therefore motivates techniques to learn better classifiers in fewer rounds.

We choose to set 70.00% as the target F1 score for FL models to reach. While substantially lower than the centrally trained classifier, this threshold nonetheless demonstrates adequate AD performance and appears sufficient to compare data augmentation algorithms to one another. As our goal is not to vastly outperform a naive implementation of FL, but rather to achieve better performance in a shorter number of rounds, we defer ensuring that FL models can indeed reach near-centralized levels to other works. Our main focus is on the period of training in which graphs of model metrics are the steepest for more apt comparisons.

IV. Data Augmentation

To mitigate the performance impact when using FL in a naive manner, we investigate the effectiveness of data augmentation. In this section, we describe statistical sampling techniques, including random oversampling, SMOTE [13], and ADASYN [14], examined in this study.

A. Random Oversampling

Dataset rebalancing via random oversampling entails repeatedly choosing a data point belonging to the minority class at random and adding it to a new dataset. This is done until a desired threshold is reached, which we set to be the amount of points in the majority class for any given client. Thus, it is possible that some points will be left out of the new rebalanced dataset and it is guaranteed that other points will

be duplicated. This has the overall effect of forcing a neural network's optimization algorithm (i.e. SGD) to pay greater attention to the minority class data to achieve a sufficiently low loss.

Random oversampling does not create any new data and it is therefore limited to the quality of the information present within the original dataset. When minority class samples are especially infrequent, this can lead to poor generalization over unseen data. Furthermore, it is unlikely that random sampling will facilitate performance metrics that would be unachievable by a naive learning algorithm when amount of minority data is not too small. Instead, the goal is to speed up the learning process and converge to a better model in fewer rounds. Adjusting hyperparameter settings such as the number of epochs can also achieve such a speedup, but this in effect duplicates majority class data as well, incurring an unnecessary time expense. Random oversampling has the advantage of only copying the infrequent data to prevent model updates from being biased towards the majority class.

B. SMOTE and ADASYN

The Synthetic Minority Oversampling Technique (SMOTE) [13] is a generative algorithm that creates new data for the underpopulated class using a point's nearest neighbors. Similar to the previous method, SMOTE will first choose a minority sample at random, but instead of reproducing it, the algorithm will select one of its k-nearest neighbors and consider a vector drawn between these two points. A value will then be chosen uniformly between 0 and 1 and be multiplied by this vector to rescale its magnitude. Finally, applying the transformed vector to the original point will indicate a new location in the feature space which will represent a synthetically generated sample.

It is worth noting that, due to the limitation of scaling the vector connecting a point and one of its neighbors by a value between 0 and 1, all synthetic samples will be confined by the boundaries of the feature space. This is a relatively cautious way of generating new points and will not result in drastic changes from the original data. Considering the similarity of kernel density estimations for each class in the Modbus dataset, such careful oversampling appears beneficial; introducing higher variance into a generative algorithm would be far more likely to cross the class boundary and unintentionally be more representative of a normal reading rather than an anomaly. SMOTE may suffer from this issue as well when class distributions become increasingly complex, but introducing new points into the training dataset can have the important benefit of better generalization to unseen data.

The Adaptive Synthetic algorithm (ADASYN) [14] 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, but in addition, it will bias the probability of selecting a point based on if it is located in a homogeneous neighborhood. Minority points that have majority examples as their closest neighbors will be chosen with greater frequency, resulting in a larger clustering of points around these regions that would ideally make boundaries more well-defined. The success of ADASYN relative to SMOTE often depends on the use case, as sparsely distributed minority data can lead to even more significant mislabeling of synthesized samples.

C. Our Augmentation Strategy

The naive FL scenario introduced in Section III-B does not employ any sampling technique but builds local models only using the data generated by the TON_IoT testbed (without any synthetically augmented instances). This may result in a biased model, especially in earlier rounds, which leads to degraded performance in the FL setting. This problem is more critical when a large number of nodes is used in training, each with 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.

We describe our augmentation strategy under the FL environment. In the typical scenario, the client (device) generates its own data. Since we assume supervised learning 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 a FL round, it analyzes the generated instances with respect to class population. In general, the number of anomalies is much smaller than normal samples, as can be seen in Table II. 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, or ADASYN. 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 duplicates instances chosen in a uniform manner. For SMOTE and ADASYN, points are also chosen uniformly, but a new synthetic anomaly is created using a randomly selected k-nearest neighbor. We set k to 5, which is a standard default value to ensure some variation without using points that are too distant in the feature space. Once each client generates the same amount of anomalies as normal samples, they will combine and shuffle the data from the two classes and begin training their local models.

V. Experiments

We conduct a series of experiments for both homogeneous and heterogeneous settings on the NERSC Cori Supercomputing system (`cori.nersc.gov`), which is configured with Intel(R) Core(TM) i5-6500 CPU @ 3.20GHz, 16 GB memory, and Ubuntu 16.04.6 LTS. The TensorFlow Federated library is used to simulate training FL models across client nodes. Again, a homogeneous setting indicates that devices generate the same amount of data, while a heterogeneous setting assumes discrepancies among devices with respect to data generation rates. We first report the experimental results observed from homogeneous settings and then discuss the results from heterogeneous settings.

A. Homogeneous Setting

For the experiments in this section, the Modbus dataset was partitioned such that each client node received an equal

TABLE IV
FL PERFORMANCE METRICS FOR VARIOUS NODES AND DATASET REBALANCING STRATEGIES OVER 100 ROUNDS

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

number of samples. Data for each client is selected at random, so this partitioning simulates data being independently and identically distributed (IID), which is a common setting for studying FL performance. The following section will consider when the IID assumption is relaxed by creating dramatic differences in the amount of data available at each client.

Trials are conducted using 5, 50, and 100 nodes in training, which result in a relatively large, medium, and 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, 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 IV, where boldface indicates the best value achieved for a fixed number of nodes. 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 two best methods after 100 rounds is extremely narrow with only a 0.06% gap in F1 score. When 100 nodes are used in training, this gap increases sharply to 9.32%.

For 50 and 100 nodes, random oversampling appears to be the best means of dataset rebalancing. This suggests that the anomalies present within the original data are sufficient for learning models, as evidenced by the centralized case which used no 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 approximately 50% testing accuracy. This indicates that the large volume of normal readings is necessary to avoid underfitting, but the issue of too few anomalies can be remedied by a simple random oversampling.

SMOTE and ADASYN did not appear to provide significant improvements to classification. Notably, Table IV 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 for more anomalies to be detected at the expense of a high 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.

Figure 3 shows F1 scores over an extended period (300 rounds) of training to compare to our target threshold of 70.00%. Our naive FL classifier attained this F1 score after 280 rounds, while it only took 211 rounds for an FL classifier using random oversampling of client data to surpass the threshold. Neither SMOTE nor ADASYN reached this performance level after 300 rounds of training and appear to be converging at a lower value. Random oversampling was able to yield a 24.6% decrease in number of rounds to attain this target with only a 6.9% 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.

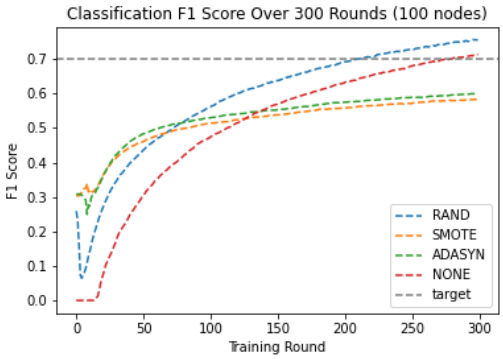
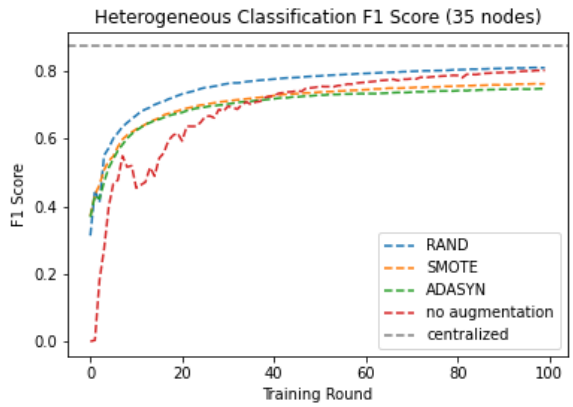


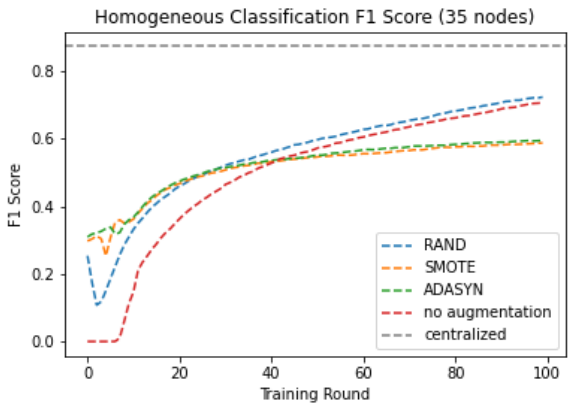
Fig. 3. F1 scores using various preprocessing methods over 100 client nodes. Random oversampling and a naive approach surpass a 70.00% target threshold while SMOTE and ADASYN do not.

B. 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 takes this into account, 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.



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



(b) Homogeneous partitioning resulting in equal client dataset sizes

Fig. 4. F1 scores for 100 rounds of training FL classifiers over 35 nodes using both heterogeneously and homogeneously partitioned data. Heterogeneous splitting results in metrics closer to those of the centralized classifier.

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. We are able to create partitionings for 35 nodes consistently such that each client has multiple anomalies to perform generative oversampling techniques.

Figure 4 shows F1 score results for an FL classifier where the number of data 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 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 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, but 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 to 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 exhibit greater clustering patterns within each class could test such an approach to determine how strong the benefit of data augmentation can be in this more extreme non-IID setting.

VI. Related Work

We provide a summary of the previous studies in the areas of IoT anomaly detection and anomaly detection over FL that are closely related to our work.

IoT Anomaly Detection. ML is a very common approach to detecting anomalies for IoT devices, but is not limited to DNNs as we employed in this work. Other studies have shown that logistic regression, support vector machines, decision trees, and random forests can reach or even surpass testing metrics of neural networks [17]. In [18], deep autoencoders have also been shown to be successful for detecting anomalous network traffic from IoT devices that have been targeted by botnet attacks. Clustering algorithms can also be suitable and efficient means of finding deviations from normal behavior as reported in [19]. Improving feature selection methods can also be useful, such as the work in [20] that did so by capitalizing on IoT-specific network behaviors.

Anomaly Detection over FL. Intrusion detection for IoT devices using FL remains an open area of research, with several novel approaches being developed such as incorporating blockchain [21] and language-analysis techniques under the assumption of edge computing [22]. Outside the scope of IoT data, malicious clients pose a significant challenge to global model convergence, so identifying and removing such erroneous updates is another important AD application [23]. In the work of [24], network AD tasks have also been studied to detect suspicious traffic flow while preserving data privacy.

VII. Conclusions and Future Work

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 upon baseline metrics in earlier rounds in both homogeneous and heterogeneous settings. Interestingly, random oversampling consistently achieves the best results over prolonged training. This appears to be due to the sparse distribution of anomalies amongst normal readings in the IoT Modbus dataset, which is likely a more realistic setting than anomalous data being highly clustered together.

Further work could be conducted by modifying data generation algorithms beyond *k*-nearest neighbors, which our results have shown has a high potential of crossing the boundary between classes. In particular, generative adversarial networks (GANs) have shown a great deal of success in synthesizing realistic new samples. However, the mode collapse problem observed when generating a wide variety of samples with diverse density functions is a technical challenge to address when using GANs and we plan to explore the effectiveness of such models to improve AD performance over FL. Another planned work is the performance improvement for semi-supervised recognition of anomalies in FL based on our preliminary observations with autoencoders showing unacceptable detection accuracy despite its benefit of the simplicity in training only with normal instances.

ACKNOWLEDGEMENT

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] Andreas P Plageras, Kostas E Psannis, Christos Stergiou, Haoxiang Wang, and Brij B Gupta. Efficient iot-based sensor big data collection–processing and analysis in smart buildings. *Future Generation Computer Systems*, 82:349–357, 2018.

[2] Elizabeth Bautista, Melissa Romanus, Thomas Davis, Cary Whitney, and Theodore 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*, page 10. ACM, 2019.

[3] Entao Luo, Md Zakirul Alam Bhuiyan, Guojun Wang, Md Arafatur Rahman, Jie Wu, and Mohammed Atiquzzaman. Privacyprotector: Privacy-protected patient data collection in iot-based healthcare systems. *IEEE Communications Magazine*, 56(2):163–168, 2018.

[4] Mohamed Elhoseny, Gustavo Ramírez-González, Osama M Abu-Elnasr, Shihab A Shawkat, N Arunkumar, and Ahmed Farouk. Secure medical data transmission model for iot-based healthcare systems. *IEEE Access*, 6:20596–20608, 2018.

[5] Angelos Mylonas, Ongun Berk Kazanci, Rune K Andersen, and Bjarne W Olesen. Capabilities and limitations of wireless co2, temperature and relative humidity sensors. *Building and Environment*, 154:362–374, 2019.

[6] Andrew Hard, Kanishka Rao, Rajiv Mathews, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604*, 2018.

[7] Koustabh Dolui, Illapha Cuba Gyllensten, Dietwig Lowet, Sam Michiels, Hans Hallez, and Danny 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, Date: 2019/06/17-2019/06/21, Location: Seoul, Korea*, 2019.

[8] Serena Zheng, Noah Apthorpe, Marshini Chetty, and Nick Feamster. User perceptions of smart home iot privacy. *Proceedings of the ACM on Human-Computer Interaction*, 2(CSCW):1–20, 2018.

[9] Huichen Lin and Neil W Bergmann. Iot privacy and security challenges for smart home environments. *Information*, 7(3):44, 2016.

[10] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282, 2017.

[11] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, H Brendan McMahan, et al. Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046*, 2019.

[12] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.

[13] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[14] Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*, pages 1322–1328. IEEE, 2008.

[15] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009.

[16] Raghavendra Chalapathy and Sanjay Chawla. Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1901.03407*, 2019.

[17] Mahmudul Hasan, Md Milon Islam, Md Ishrak Islam Zarif, and MMA Hashem. Attack and anomaly detection in iot sensors in iot sites using machine learning approaches. *Internet of Things*, 7:100059, 2019.

[18] Yair Meidan, Michael Bohadana, Yael Mathov, Yisroel Mirsky, Asaf Shabtai, Dominik Breitenbacher, and Yuval Elovici. N-baiot—network-based detection of iot botnet attacks using deep autoencoders. *IEEE Pervasive Computing*, 17(3):12–22, 2018.

[19] Lingjuan Lyu, Jiong Jin, Sutharshan Rajasegarar, Xuanli He, and Marimuthu Palaniswami. Fog-empowered anomaly detection in iot using hyperellipsoidal clustering. *IEEE Internet of Things Journal*, 4(5):1174–1184, 2017.

[20] Rohan Doshi, Noah Apthorpe, and Nick Feamster. Machine learning ddos detection for consumer internet of things devices. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 29–35. IEEE, 2018.

[21] Davy Preuveneers, Vera Rimmer, Ilias Tsingenopoulos, Jan Spooren, Wouter Joosen, and Elisabeth Ilie-Zudor. Chained anomaly detection models for federated learning: An intrusion detection case study. *Applied Sciences*, 8(12):2663, 2018.

[22] Thien Duc Nguyen, Samuel Marchal, Markus Miettinen, Hossein Fereidooni, N Asokan, and Ahmad-Reza Sadeghi. Dïot: A federated self-learning anomaly detection system for iot. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 756–767. IEEE, 2019.

[23] Suyi Li, Yong Cheng, Wei Wang, Yang Liu, and Tianjian Chen. Learning to detect malicious clients for robust federated learning. *arXiv preprint arXiv:2002.00211*, 2020.

[24] Ying Zhao, Junjun Chen, Di Wu, Jian Teng, and Shui Yu. Multi-task network anomaly detection using federated learning. In *Proceedings of the Tenth International Symposium on Information and Communication Technology*, pages 273–279, 2019.