Disparity-Aware Federated Learning for Intrusion Detection Systems in Imbalanced Non-IID Settings

Md Mohaiminul Islam
mohaiminmahim@gmail.com
Bangladesh University of Engineering and Technology
Dhaka, Bangladesh

A. B. M. Alim Al Islam
alim_razi@cse.buet.ac.bd
Bangladesh University of Engineering and Technology
Dhaka, Bangladesh

ABSTRACT

Many variants of Federated Learning have been proposed to settle different challenges that come with numerous practical applications, one of which is dealing with non-IID data sources. As decentralized data sources in real life are bound to be non-IID, this is one of the hardest challenges, and yet the earliest federated algorithms struggle to resolve this issue, resulting in worse non-IID performance. Also, applications that require capturing really intricate insights from data while upholding the latest data privacy standards, such as Intrusion Detection Systems (IDS) have enabled the use of FL in those domains. In this article, we propose a novel Disparity-Aware federated learning approach that tackles non-IID and data imbalance from both global and local levels of FL. Unlike some previous works that require data sharing, our approach does not impose any considerable communication overhead and renders itself applicable even in asynchronous federated learning schemes. Combining state-of-the-art imbalanced learning techniques at the client end and a fast and smart aggregation scheme that considers local data distribution scenarios at the server end, this approach improves on the performance of traditional deep learning methods. Experimental evaluation also shows improvement over earlier popular FL algorithms. We establish our approach by testing in the wild using complicated datasets for Intrusion Detection Systems.

KEYWORDS

Federated Learning, Intrusion Detection, Disparity-awareness, Imbalanced Learning, Non-IID, NSL-KDD, UNSW-NB15

1 INTRODUCTION

Distributed systems play a central role in many recent technological advancements. The recent surge in consumer-based software and smart devices has prompted corporations with billions of users to adopt distributed architectures for their networking and data warehousing needs. Despite these developments, cyber threats have also significantly grown proportionally. Attackers often dismantle networks of millions of devices, leading to widespread denial of service and data leaks. Therefore, rapidly detecting intrusions in modern protocols is a significant task and an open-ended challenge. Machine learning approaches, especially deep-learning-based ones have been recognized as the go-to method for developing intrusion detection mechanisms. Such tasks demand large quantities of data which in practice only be attained through continuous monitoring of many devices over a large network, e.g., a network of interconnecting IOT devices. But to collect data from a large network poses a notable challenge: collecting users' private network traffic data and centralizing it to train said, deep learning model. An article by authors in [1] discovered that a high packet loss rate negatively impacts prediction accuracy as network size increases. Consequently, gathering such vast volumes of data is almost impractical for modern communication protocols, not to mention maintaining data privacy standards requires extensive effort if even plausible at all. Federated Learning is an already well-established framework for the decentralized training of large datasets. Past works proposed federated learning for distributed malware detection[2] techniques as well as privacy-preserving[3, 4] computing.

Although a significant volume of work has tried to reconcile differentially private computing with distributed intrusion detection models, challenges appear when applying these methodologies in practice, such as reducing communication overhead and making federated architectures viable for real-time applications [5, 6], mitigating adversarial security threats, (e.g., federated poisoning attacks [7]), developing a robust architecture that excels in a *Non-IID (Non-identically and independently distributed)* client environment [8]. Designing and deploying federated solutions on low-power computational devices [9, 10].

In this paper, we aim to provide a solution for client environments that deal with heterogeneous and highly imbalanced data. Previous works in this domain mainly follow one of two themes: either reconciling the divergence in client models through techniques such as data sharing or regularizing updates, etc. or clustering the clients based upon their update similarity to mitigate the dominance of the majority classes. Our approach tackles non-IIDness and class imbalance from both local and global scales. Moreover, we introduce *disparity awareness* in federated learning, which takes into consideration the ever-changing data landscape of local devices, as that is almost always the case for modern distributed learning systems. Many large corporations rely on the collection of client data and incremental fusion of new data to train and improve their machine-learning models. Our learning algorithm acknowledges this dynamic scenario and adjusts the hyperparameters accordingly. Moreover, in this method, clients send their own *Imbalance ratio* to the server as a client imbalance vector that represents their current data distribution as a whole. In other words, the server is made aware of the class disparity of each client's local data, which it considers while aggregating the subsequent updates. No significant computational overhead is imposed on the server. We empirically test our method on a complex scenario (i.e., intrusion detection) using a realistic benchmark dataset in order to validate it. To summarize, our key contributions are:

- Proposing a novel scheme Disparity-Aware Federated Learning (DAFL) to mitigate federated learning performance deterioration in highly imbalanced and/or non-IID scenarios.

- We build and test a system on top of a traditional deep-learning system to ensure and verify backward compatibility with existing systems.
- To consolidate our findings, our system running on *DAFL* aggregation scheme is tested with benchmark datasets such as *NSL-KDD* and *UNSW-NB15*.

Previous studies on Federated Learning in non-IID scenarios have often failed to capture the complexity of data in real-life applications such as IDS systems. Consequently, a comprehensive study that addresses these challenges using benchmark datasets is lacking in this field. Our goal is to bridge that gap through this work.

2 RELATED WORK

The scenario originally envisioned by [11] describes the decentralized optimization issue in distributed-iid environments such as large-scale data centers. Although their *FedAVG.* algorithm served as a powerful baseline, but it is not robust enough to handle non-IID data[12]. Many surveys [13, 14] covered the recent developments and challenges [15] in that domain. The work of [16] introduces two new aggregation functions that are robust and are based on the coordinate-wise median and the coordinate-wise trimmed mean of the models sent by the clients. In [17] the authors proposed resampling to reduce heterogeneity in the distribution of the models sent by the clients. It is meant to be applied before using a robust aggregation function, and it aims at reducing the side effects that such a function has when applied to models trained with non-IID datasets. Zhao et al., [18] proposed that the global agent share a small subset of public data with heterogeneous clients to prevent their updates from straying too far from the global model, thereby increasing training stability and bolstering robust performance. Although this has the prerequisite of such public data being sufficiently available. The work of [19], solves the same issue by adding a regularization term to the local optimization, Which limits the weight divergence between local and global models. Both approaches result in a single global model, which aims to improve accuracy. However, under some non-IID settings (e.g., clients with conflicting optimization goals), a single joint global model that is optimal for all clients is not possible [20]. Ghosh et al., [21] use K-means to cluster client updates for the purpose of identifying and isolating byzantine clusters of clients prior to the averaging step. Previous works can mostly be divided into two categories regarding their approach to resolving non-IID data. The first of which aims to personalize federated learning in order to fine-tune the learning process at the local level. Such works were summarized by [22] among which transfer learning[23, 24] and meta[25] learning processes were established to be most successful. On the other hand, client clustering approaches[8, 25] tend to group similar clients together based on their update likeness (e.g., cosine similarity), which covers a wide range of non-IID settings. Castellon et al.,(2022) [26] proposed incremental clustering of temporally independent client updates for better results.

3 PROPOSED METHODOLOGY

To demonstrate the full effect of our proposed methodology, we consider the scenario of a homogenous distributed system of edge devices where federated learning can be performed using the data acquired at individual nodes. All the smart edge devices usually request data from a cloud server, ensuring periodic communication of each node with a central entity fitting for the role of a central aggregation server in a federated learning environment. In this scenario, often the cloud server initiates communication with the edge devices, usually in the form of software improvements or updates. Such updates can carry the new learning parameters of our FL scheme. This bidirectional, periodic communication makes our envisioned system eligible for the proposed framework. The system elements are described as follows:

Edge/Client devices: These entities generate their own data and preprocess it individually. We can also expect that the clients are unwilling to share their data to preserve the privacy and safety of their devices and enforce safeguards against security risks such as exposure of their data.

Central Server: Acting as an intermediary between the clients if any communication ever occurs, also assuming a central role in our learning algorithm. Also, the central server sends out updates to improve the performance of the whole ecosystem as well as ensure model and client data safety.

The system consists of a set of N client workers which we will refer to as local agents, each acquire, preprocess, and build their own dataset D^k of approximately equal size m and d dimensions. In real life, each $D^k{}_{,k\in\{1\ldots N\}}$ also might be geographically separated and together make up the total federated dataset $D = \bigcup_{k\in\{1..N\}} D^k$. The common goal of all the local agents is to cooperate with a global agent and train an IDS System that learns from all the clients' data without compromising their privacy (i.e., sharing data with the global agent). In practice, each of the local datasets D^n is most likely to be Non-IID. Usually for a supervised learning task, assume each data record $(\mathbf{x}, y)$, where x is the feature vector and y is the known label, follows a certain distribution $P_k(\mathbf{x}, y)$. Non-IIDness can be defined as a setting where different clients have a different P_k for their local datasets[12]. This property can primarily be classified into attribute and label distribution skew[27]. We can classify a federated dataset to be label skewed if the conditional feature distribution $P_k(x|y)$ is agreed upon among the clients but the label distribution $P_k(y)$ varies from one client to the other. Label skew can be further subdivided into *Label size imbalance* and *Label distribution imbalance*. We chose two datasets: *NSL-KDD, UNSW-NB15* for our experiments, which are inherently imbalanced in terms of label distribution. Data expresses Non-IID behavior and also can be expressed through attribute skew, which refers to the variation of $P_k(x)$ for each client. This skew of distribution can be either non-overlapping, partially overlapping, or fully overlapping.

3.1 Disparity Awareness in FL

This section describes a robust framework for federated learning that has been used in the experimental section on benchmark datasets for intrusion detection systems.

In the conventional Federated Learning, if we denote the model parameters on the ith device after t iterations as w_i^t, the federated averaging step can be represented as follows:

$$w^{t+1} = \frac{\sum_{i=1}^{N} n_i w_i^t}{\sum_{i=1}^{N} n_i}$$

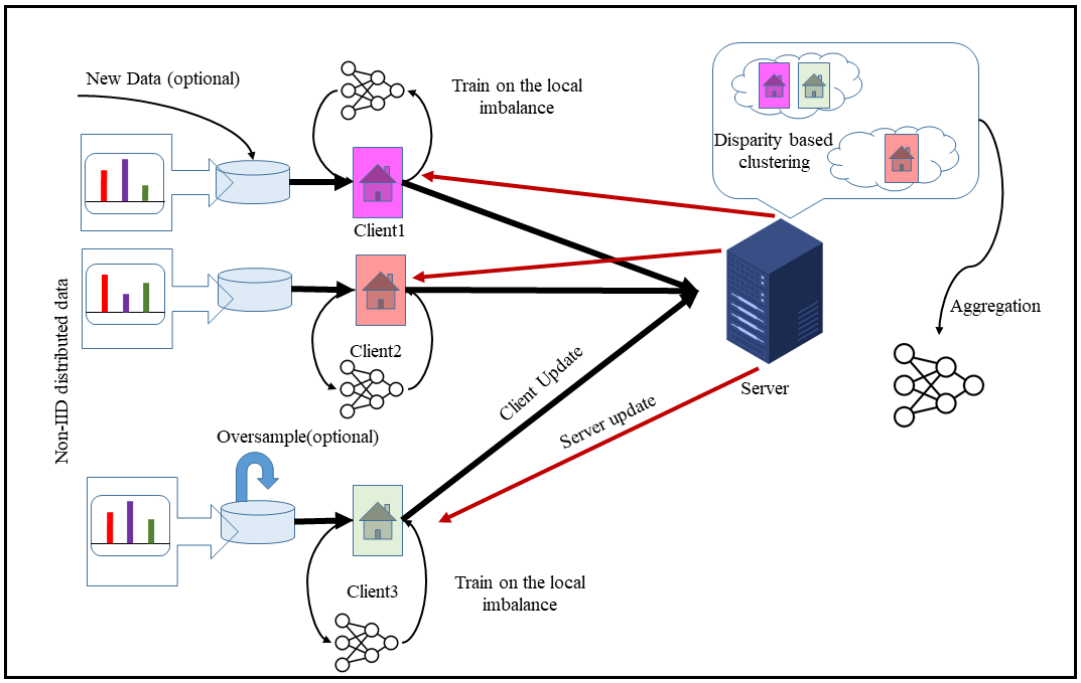

Figure 1: Workflow of disparity aware federated learning.

Here, N represents the number of participating devices, and n_i is the number of data samples on the i-th device. The aggregated model w^{t+1} is then distributed back to the devices for the next round of local computations and updates. Our proposed federated infrastructure, which aims to collectively build a distributed Intrusion Detection System (IDS), incorporates a similar process but instead uses a weighted FedAVG. algorithm. To consider class distribution disparity at the client level, we train the local model on a customized Loss function, similar to the Unified Focal Loss developed by Yeung et al., (2022) [28], which combines distribution-based losses such as the modified asymmetric Focal Loss ($\mathcal{L}_{maF}$) and region-based loss like modified asymmetric Focal Tversky loss ($\mathcal{L}_{maFT}$). The asymmetric variation of Unified Focal loss can be depicted as:

$$\mathcal{L}_{aUF} = \lambda \mathcal{L}_{maF} + (1 - \lambda)\mathcal{L}_{maFT} \tag{1}$$

Here $\lambda \in [0, 1]$ is a hyperparameter that balances the two losses. Originally developed for object detection and image segmentation tasks, The Focal loss[29] suppresses the dominant class (or the background class for image segmentation), which in turn also suppresses the minority classes as the focal parameter is applied to all the classes. Enabling asymmetry enhances the suppression of the dominant class more than the rare class. On the other hand, in the modified Focal Tversky loss[30], the focal parameter is removed, retaining only the enhancement of minority classes. We decide to set $\lambda = 0.7$. The intuition is to suppress the dominant classes more and not enhance the rare classes too much in order to avoid overall performance deterioration. For other hyperparameters, δ is calculated from the imbalance vector (described in 3.2) as the inverse of the corresponding relative distribution of each class, and γ is set to the recommended default of 0.5.

At the server end, a hierarchical grouping is employed inspired by Briggs et al., (2020) [8]. The point of difference is that the prior work clusters clients based on their updates, whereas ours performs the clustering based on their data distribution. This clustering is performed by calculating the similarity between client imbalance vectors. Essentially, the clients send their imbalance vectors after each round of training, and the cosine similarity of these vectors is used to group clients. The average update of each group is calculated, and the group updates are aggregated to nullify the influence of the majority of updates. For highly imbalanced scenarios, some, if not the majority, of clients receive data without any samples from the majority class. Hence, client-level measures against imbalance are rendered virtually useless. The Clients can also choose to synthetically upsample or infuse new data into the learning system. As the imbalance vectors are propagated alongside each update, any changes in the client's data distribution are made aware to the server, in turn altering the client groups for future aggregation. Our framework involves a central and a local model. The architecture and hyperparameters of the central model, such as learning rates for the server α_g and the client α_l, batch size b, are initially selected. This information is then broadcast to the active devices in the network, and the subsequent process of localized learning, updates, and global model refinement takes place similarly to the aforementioned scheme. It's important to note that this iterative process allows the models to learn from decentralized data sources while upholding data privacy and security regulations. It is also experimentally validated that in specific scenarios such learning algorithms converge in a relatively lower number of rounds while

Table 1: List of Notations

Symbol	Description
n	number of total clients.
n_i	number of participating clients in the i-th comm. round.
D^k	Local dataset for i-th client.
$\theta_{g,k}$	Global model parameters for i-th round.
$\theta_{l,k,i}$	Local model parameters of k-th client for i-th round.
V_k	Imbalance vector sent by kth client.
C_i	Subset of clients selected at ith round.
$\Delta\theta_{l,k,i}$	Local model update of k-th client for the i-th comm round.
$\Delta\theta_{agg.,i}$	Weighted average of local model updates for i-th round.
α_g, α_l	Learning rates for global and local models respectively.
$\mathcal{L}_{aUF}$	Asymmetric unified focal loss.
η	batch-wise gradient calculation function.
T	Total number of training rounds.

reflecting the collective knowledge of all participating devices, enhancing the overall system's performance and accuracy. The architecture for our local and global models used in the experimental section follows the architecture of DNN-3 [31], which is a deep neural network with three hidden layers. Layerwise details are laid out in table 2. We aim to develop a multiclass classifier, hence, the input and output shapes vary, respectively depending on the feature vector size and the number of label classes of the corresponding dataset being used. For our experiments with NSL-KDD, the shapes of the input and output layers were 53 and five, respectively. We used *Adam* optimizer for both client and server, which is based on adaptive estimation of first- and second-order moments. We went for a homogeneous architecture for the server-client system to eliminate model-specific results and focus on the efficacy and applicability of our DAFL algorithm in the case of IDS systems.

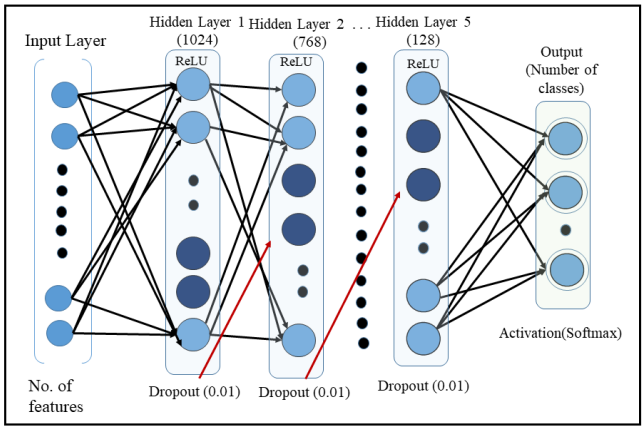

Figure 2: Model Architecture.

3.2 Disparity-Aware Federated Learning Algorithm

We now describe the details and intuition behind ***Disparity-Aware Federated Learning*(DAFL)** algorithm. The notations used in the algorithmic discussion are briefly described in table 1. Algorithm 1 represents the work done from the client side on each of the global communication rounds. The main difference between the *ClientUpdate* procedure with the original FedAvg. is, the clients are responsible for sending an *imbalance vector*,$V_K = [r_1..., r_c]$ that is co-related to their local data sistribution. We can imagine this vector to be as simple as the inverse of each class's relative frequency in the client's data compared to the major class. For example, if some client is working with three classes and has 50%,25%, and 25% data of the three classes respectively the $V_k = [1, 0.5, 0.5]$. The clients also have the liberty to upsample their local batched data to deal with label distribution skew. As a new set of class weights is transmitted with each *ClientUpdate*, irrespective of the choice oversampling, the central server can collect and aggregate the parameters received from the clients accordingly. Moreover, clients use asymmetric unified focal loss described in the previous section to deal with class disparity on a local level.

Algorithm 1: Client k in i-th global communication round

Input:, Receive $\theta_{g,i-1}$ from server.
Load data shard $d_k \subset D^k$ from storage.
Output: updated parameter set $\theta_{l,k,i}$, w_k
procedure ClientUpdate:
Initialize M_l
$\theta_{g,k,i-1} \leftarrow \theta_{l,k,i}$
calculate $V_k \leftarrow f(d_k)$
Loss hyperparameter, $\delta \leftarrow V_k^{-1}$
for $t = 1$ to NUM_ROUNDS **do**
  for each batch b in d_k **do**
    optimize loss $loss \leftarrow Adam(\mathcal{L}_{aUF}(\theta_{l,k,i}; \delta; \lambda; \gamma))$
    $\theta_{l,k,i} \leftarrow \theta_{l,k,i-1} - \alpha_l * \eta(\Delta\theta_{l,k,i-1}; b)$
  end for
end for
Send $\theta_{l,k,i}, V_k$ to server
end procedure

Algorithm 2 depicts the computation taking place at the server end throughout the learning process. The algorithm employs a global model architecture M_l and learning rates α_g, α_l to facilitate collaborative model training among a network of clients. The algorithm focuses on cluster-based aggregation, where the groups are from the disparity vectors sent by different clients. The process begins with the initialization of the global and local learning parameters. Clients are then provided with the global model architecture and a learning rate. In the main execution loop, spanning several rounds, a subset of clients is selected, and each participating client updates its local model based on the global model's parameters and the provided learning rate. The central server receives each client update and groups them based on the cosine similarity of the imbalance vectors received. For two vectors V_i and V_j the cosine similarity can be formulated as:

$$\cos(\mathbf{V_i} \cdot \mathbf{V_j}) = \frac{\mathbf{V_i V_j}}{\|\mathbf{V_i}\|\|\mathbf{V_j}\|} = \frac{\sum_{k=1}^{C} \mathbf{V_{ik}} \mathbf{V_{jk}}}{\sqrt{\sum_{i=1}^{C} (\mathbf{V_{ik}})^2} \sqrt{\sum_{i=1}^{C} (\mathbf{V_{jk}})^2}} \quad (2)$$

For each cluster, the average update is calculated and the averages of all the groups are aggregated using weighted averaging.

Algorithm 2: Weight-based aggregation in Central server

Input:
Global Model architecture M_l
Learning rates α_g, α_l
Output: Global model M_l and finalized weights $\theta_{g,T}$
procedure ServerExecution:
Initialize $\theta_{g,0}$
for each Client $c \in C$ **do**
  Send M_l, α_l to clients
end for
for $i = 1$ to T **do**
  Select random client set $C_i \subseteq C$ to for round i.
  for each client $c_k \in C_i$ **do**
    Send $\theta_{g,i-1}, \alpha_l$ to c_k
    $V_k, \theta_{l,k,i} \leftarrow ClientUpdate(\theta_{g,i-1}; \alpha_l)$
    calculate similarity of V_k with other clients and insert into cluster G_j.
  end for
  for each client cluster $G_j \in G$ **do**
    $\theta_{l,j} \leftarrow Average(\theta_{l,k,i})$, where client $c_k \in G_j$
  end for
  Aggregate all the cluster updates, $\theta_{g,i} \leftarrow Aggregate(\theta_{l,j})$
  load $\theta_{g,i}$ into M_l
  Send updated $\theta_{g,i}$ to all clients for next round
end for
end procedure

4 DATASETS AND PRE-PROCESSING

We used two benchmark datasets for intrusion detection in this study. These datasets were pre-processed and converted into federated datasets using the TensorFlow federated learning framework. The following section describes the composition of the datasets and the pre-processing steps.

4.1 Dataset description

4.1.1 NSL-KDD. The dataset used to run the first batch of experiments was the NSL-KDD, which has been used profusely in training intrusion detection systems both to recognize benign and malicious traffic. NSL-KDD was designed to facilitate the development and evaluation of intrusion detection systems (IDS) that can effectively distinguish between normal network traffic and various types of network attacks. After its public release in 2009, numerous works from different domains have used it for developing and testing machine learning-based IDS systems [32–35]. It contains data on 22 different varieties of attacks as well as normal traffic. Also, These attacks include different categories such as denial-of-service (DoS), probing, user-to-root (U2R), and remote-to-local (R2L) attacks, encompassing a wide range of potential security threats. NSL-KDD has a separate training set and testing set. The training set consists of around 125,000 records, while the testing set contains about 23,500 records. Both sets include a mix of normal and attack connections. It has 41 features including categorical and numerical ones.

4.1.2 UNSW-NB15. This provides one of the most comprehensive and realistic representations of network traffic and attacks, and therefore, is widely used to train and evaluate different classes of IDS systems. The dataset is created from real-world network traffic collected in a controlled environment. It was publicly released by Mustafa et al.,(2015)[36]. The dataset comprises a total of around 2.5 million records. This dataset has been used in various scopes including but not limited to rule-based IDS[37], deep learning IDS systems[38], and even nonlinear feed-forward networks[39]. It was also studied in the scope of federated learning-based IDS systems, e.g., Federated GAN networks[40], federated semi-supervised IDS[41]. For our simulation purposes, we used the filtered training subset containing nearly 83,000 records of various types. The UNSW-NB15 dataset encompasses nine different attack categories. Our training set contains all of these labels, the distribution of which is shown in figure 4. The attack classifications within the UNSW-NB15 dataset remain consistent, and we have evaluated its performance using a multiclass classification scenario involving ten different labels.

4.2 Exploring the Non-IID nature of data

To ensure the label distribution imbalance we can visualize the relative abundance of different classes in our datasets. For NSL-KDD we relabeled the dataset into five major classes, i.e., DoS, Probe, U2R, and R2L attacks and normal records, to improve model performance and benchmark against other related works. In figure 3 and 4 we can see the relative abundance of different classes in the training and testing sets of NSL-KDD and UNSW-NB15 respectively. The training set is randomly distributed among the clients to ensure the non-IIDness between them.

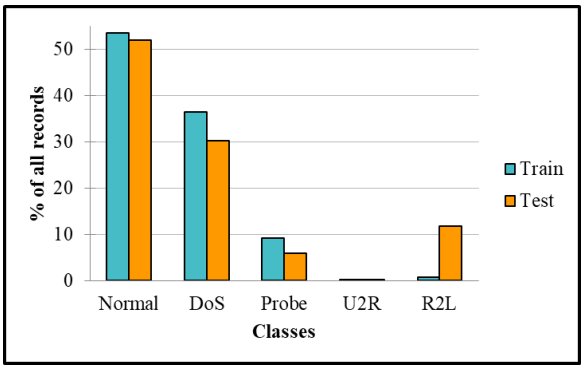

Figure 3: Relative abundance of attacks in NSL-KDD dataset.

In preprocessing step 3, we remove the labels from categorical features that are seldom used. We only do this with features of extremely high cardinality. This ensures a partial overlapping attribute skew in our data as discussed in section 3. To summarize, the overall analysis proves both of the chosen datasets are suitable for this study.

4.3 Data preprocessing

4.3.1 Data Relabeling. For NSL-KDD we relabeled the attack data into five generic classes as mentioned previously. This increases model performance and makes it easy to compare the results with

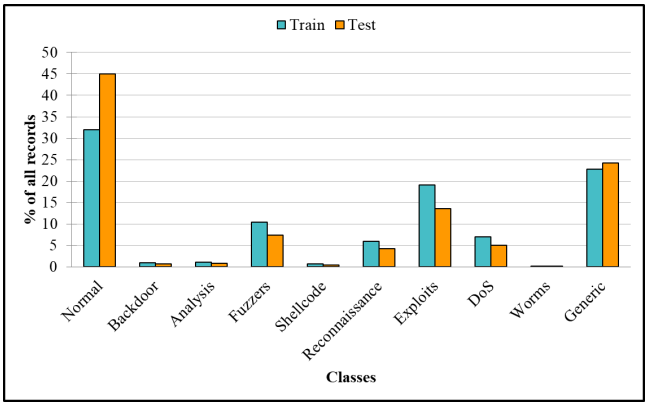

Figure 4: Relative abundance of attacks in UNSW-NB15 dataset.

preexisting work based on both the NSL-KDD and KDD '99 datasets. The labels for the UNSW-NB15 dataset remain unchanged.

4.3.2 Pruning outliers. In order to remove extreme values and outliers we clamp the data and prune feature values above 10× the median value for that feature. Extreme values were reset to the $99.5 - th$ percentile. The factor or 10× was randomly selected to avoid bimodal and small value distributions from being excessively pruned. This preprocessing technique was used for both datasets.

4.3.3 Log-normalization. Some of the features with a continuous distribution and extreme values skew the distribution too much. Therefore, continuous features were separated and log-normalized for both datasets. This drastically improves the performance of optimizer functions.

4.3.4 Reducing categorical labels. Some categorical features can have numerous discrete values. In the next preprocessing step, while one-hot encoding is applied, these features turn out to be very troublesome as encoding them explodes the dimensionality of the overall feature space. That is why in this step we reduced the number of discrete values that a categorical feature can have. We only apply this for features with more than ten different labels. To achieve this, first, all the label values were sorted by their frequency, and the ten most frequent labels remained, the rest were replaced with don't cares.

4.3.5 Encoding non-numeric features. One-hot encoding is a technique employed in data preprocessing to transform categorical attributes into numerical ones. This approach is applicable for features such as *protocol_type* alone; it can be extended to other attributes like *service* and *flag*. Through this process, categorical data is transformed into a format that is suitable for various machine-learning algorithms and analytical tasks.

4.3.6 standardizing numerical features. Finally, we apply min-max scaling to all the numerical features. This scaling ensures that the features are brought to a common scale, which eliminates the influence of the measurement unit on the model training and makes the training result more dependent on the characteristics of the data itself.

5 EXPERIMENTAL EVALUATION

We performed multiple sets of experiments on both of the datasets. Two experimental setups were used. In the first one, a Deep neural network was trained to perform binary and multiclass classification in two different test suits for NSL-KDD and binary classification for UNSW-NB15. No federated learning was employed in this setup. In the second experimental setup, we simulated a network of clients collaboratively learning by implementing our DAFL algorithm. The same suit of tests was completed for this setup as well and after that, we compiled and compared the two setups to study the efficacy of *Disparity-Aware Federated Learning* process. We evaluated both setups in terms of training accuracy, validation loss, precision, recall, F1-score, and overall macro testing accuracy and summarized the results in the following subsections.

5.1 IDS without FL

This setup is designed to be the control for testing our federated learning algorithm. We kept the model architecture quite similar to our local model in the federated learning setup. We trained this model separately to perform binary and multiclass classification. For the NSL-KDD dataset, the number of output labels was relabeled into five major classes for multiclass classification, and for binary classification, all the attacks were relabeled into the same class, leaving behind only the benign records. For both cases, we can visualize training loss as the training progresses in figure 5. The model seemed to converge quickly and perform quite well on all of our evaluation metrics. Details of the evaluation are summarized in the table 2 and 3.

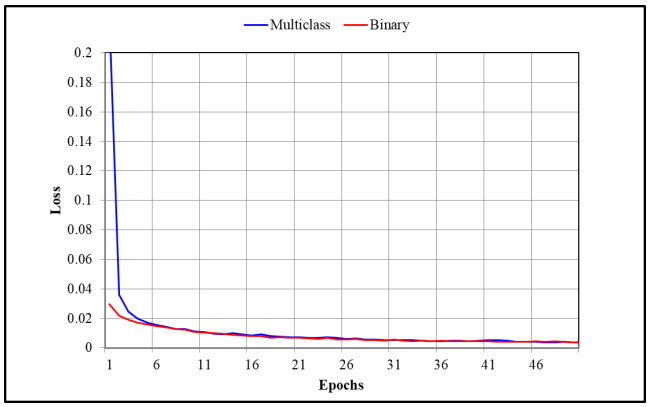

Figure 5: Validation loss for binary and multiclass classifications for DL IDS.

5.2 DA-Federated IDS

We built a collaborative distributed learning network as described in section 3. For simulation purposes, we fixed the number of clients to n = 10 and let all the clients choose to randomly oversample their data (to simulate the feeding of new data into the network). From 6 and 7 we can see the average loss and accuracy of our model which suggests it nears convergence within a relatively low number of comm rounds. We can also see fluctuations that indicate the effect of large penalties imposed by our loss function for misclassifying minority classes. We can also see from table 2 and 3 that our system had improved in both binary and multiclass classification tasks after implementing Disparity-Aware Federated Learning in both datasets. Table 2 and 3 show the comparison between our disparity-aware federated setup and other deep learning models trained on

globally oversampled data with categorical cross-entropy as the loss function. The deep learning model trained on NSL-KDD closely resembles the local model architecture of our clients. On the other hand, UNSW-NB15 performance data was compared to the data acquired from [42], in which they developed an LSTM-based DNN on three combined datasets (including UNSW-NB15).

We can see an average accuracy improvement of 2.8 ± 0.38% for binary and 7.12 ± 1.18% for multiclass classification respectively. As most of the records belong to the relatively abundant classes the more significant metric to improve here is the F1 score, which improved 6.31% for binary and 7.47% for multiclass classification on average over the two datasets combined. This indicates the desired improvement in the classification of marginal or minority classes. Precision and recall Both also improved over 2.5% and 7.5% approximately which also strengthens our assertion.

Table 4 compares our gains with some of the various federated algorithms. The table shows Federated-GAN is closest to us in performance gains, But in this work, every client generates synthetic data points using generative adversarial networks and advanced image processing, Whereas, feeding new data or oversampling is optional for our clients.

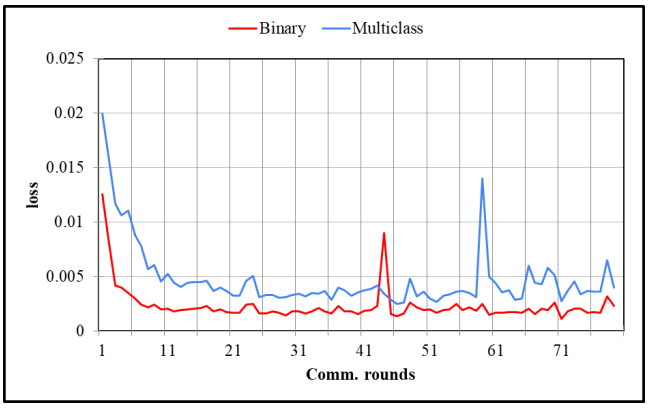

Figure 6: Validation loss for binary and multiclass classifications for DA-Fed IDS.

6 CONCLUSION

In this work, we have shown our federated learning algorithm can supersede the traditional ML approach in both training time and performance. Our experiments also show promising results for highly imbalanced datasets. Moreover, all our experimentation was performed on realistic benchmark datasets for IDS systems. In the future, we look to fine-tune and improve the algorithm to tackle other Non-IID attributes such as attribute skew that could have huge implications in developing large scaled generalized heterogeneous federated learning systems.

REFERENCES

[1] F. Tang, B. Mao, Z. Fadlullah, and N. Kato. On a novel deep-learning-based intelligent partially overlapping channel assignment in sdn-iot. *IEEE Communications Magazine*, 56:80–86, 2018.

[2] R. Taheri, M. Shojafar, M. Alazab, and R. FED-IIoT: A Tafazolli. robust federated malware detection architecture in industrial iot. *IEEE Transactions On Industrial Informatics*, 17:8442–8452, 2020.

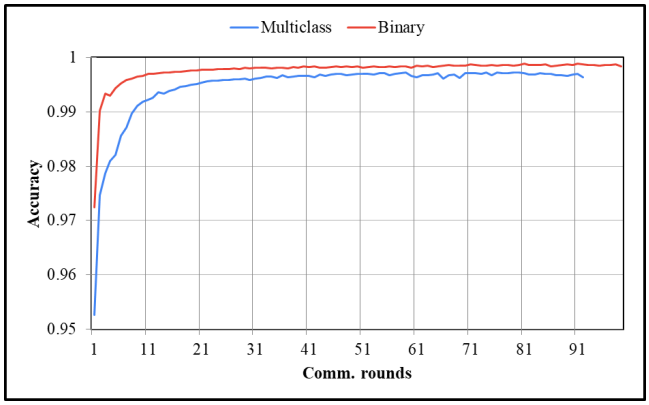

Figure 7: Training accuracy for binary and multiclass classifications for DA-Fed IDS.

[3] Y. Cheng, Y. Liu, T. Chen, and Q. Yang. Federated learning for privacy-preserving ai. *Communications Of The ACM*, 63:33–36, 2020.

[4] M. Gong, Y. Xie, K. Pan, K. Feng, and A. A Qin. Survey on differentially private machine learning [review article]. *IEEE Computational Intelligence Magazine*, 15:49–64, 2020.

[5] Q. Qin, K. Poularakis, K. Leung, and L. Tassiulas. Line-speed and scalable intrusion detection at the network edge via federated learning. *2020 IFIP Networking Conference (Networking)*, pages 352–360, 2020.

[6] Y. Chen, Y. Ning, M. Slawski, and H. Rangwala. Asynchronous online federated learning for edge devices with non-iid data. *2020 IEEE International Conference On Big Data (Big Data)*, pages 15–24, 2020.

[7] A. Fuller, Z. Fan, C. Day, and C. Digital Twin: Enabling Technologies Barlow. Challenges and open research. *IEEE Access*, 8:8952–10897, 2020.

[8] C. Briggs, Z. Fan, and P. Andras. Federated learning with hierarchical clustering of local updates to improve training on non-iid data. *2020 International Joint Conference On Neural Networks (IJCNN)*, pages 1–9, 2020.

[9] Y. Han, D. Li, H. Qi, J. Ren, and X. Wang. Federated learning-based computation offloading optimization in edge computing-supported internet of things. *Proceedings Of The ACM Turing Celebration Conference-China*, pages 1–5, 2019.

[10] S. Latif, Z. Zou, Z. Idrees, and J. A Ahmad. Novel attack detection scheme for the industrial internet of things using a lightweight random neural network. *IEEE Access*, 8:89337–89350, 2020.

[11] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Arcas. Communication-efficient learning of deep networks from decentralized data. *Proceedings Of The 20th International Conference On Artificial Intelligence And Statistics*, 54:1273–1282.

[12] Yaochu Jin, Hangyu Zhu, Jinjin Xu, and Yang Chen. *Federated Learning: Fundamentals and Advances.* Springer Nature, 2022.

[13] Q. Yang, Y. Liu, T. Chen, and Y. Federated Machine Learning: Concept and Tong. Applications. *ACM Trans. Intell. Syst. Technol*, 10.

[14] P. Kairouz, H. McMahan, B. Avent, A. Bellet, M. Bennis, A. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, and Others Advances. and open problems in federated learning. *Foundations And Trends® In Machine Learning*, 14:1–210, 2021.

[15] S. Agrawal, S. Sarkar, O. Aouedi, G. Yenduri, K. Piamrat, M. Alazab, S. Bhattacharya, P. Maddikunta, and T. Gadekallu. Federated learning for intrusion detection system: Concepts, challenges and future directions. *Computer Communications*, 195:346–361, 2022.

[16] D. Yin, Y. Chen, R. Kannan, and P. Byzantine-Robust Distributed Learning Bartlett. Towards optimal statistical rates. *Proceedings Of The 35th International Conference On Machine Learning*, 80:5650–5659.

[17] S. Karimireddy, L. He, and M. Jaggi. Technical report, *ArXiv*, title = Byzantine-robust learning on heterogeneous datasets via bucketing, type = Preprint, year = 2020, archivePrefix = arXiv, eprint = 2006.09365.

[18] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra. Technical report, *ArXiv*, title = Federated learning with non-iid data, type = Preprint, year = 2018, archivePrefix = arXiv, eprint = 1806.00582.

[19] T. Li, A. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith. Federated optimization in heterogeneous networks. *Proceedings Of Machine Learning And Systems*, 2:429–450, 2020.

[20] F. Sattler, K. M"uller, and W. Clustered federated learning Samek. Model-agnostic distributed multitask optimization under privacy constraints. *IEEE Transactions On Neural Networks And Learning Systems*, 32:3710–3722, 2020.

Table 2: Comparing traditional DL and DAFederated learning setup in both binary and multiclass classification of NSL-KDD dataset.

Evaluation Criteria	NSL-KDD (Binary)		NSL-KDD (Multiclass)	
	DNN-3[31]	DAFed IDS	DNN-3[31]	DAFed IDS
Accuracy	92.78%	95.19%	77.60%	85.89%
Precision	80.41%	94.22%	81.38%	91.35%
Recall	75.55%	81.48%	77.81%	76.79%
F1-Score	76.38%	87.30%	72.72%	83.48%

Table 3: Comparing traditional ML and DAFederated learning setup in both binary and multiclass classification of UNSW-NB15 dataset.

Evaluation Criteria	UNSW-NB15 (Binary)		UNSW-NB15 (Multiclass)	
	FastRNN[42]	DAFed IDS	FastRNN[42]	DAFed IDS
Accuracy	95.53%	98.71%	82.62%	88.56%
Precision	94.43%	97.53%	82.88%	86.16%
Recall	95.94%	97.55%	82.62%	88.56%
F1-Score	95.39%	97.25%	81.31%	87.71%

Table 4: Improvement over FedAvg. for ours and some other federated learning approaches.

Algorithm	Acc. (improvement)	F1-Score (improvement)
HCFL [8]	2.8%	-
FedProx [19]	0.68%	-
FedBe [43]	3.01%	-
FedEnsemble [44]	5.26%	-
Fed-GAN [45]	10.5%	9.02%
DAFL (Ours)	8.29%	10.89%

[21] A. Ghosh, J. Hong, D. Yin, and K. Ramchandran. Technical report, *ArXiv*, title = Robust federated learning in a heterogeneous environment, type = Preprint, year = 2019, archivePrefix = arXiv, eprint = 1906.06629.

[22] Alysa Ziying Tan, Han Yu, Lizhen Cui, and Qiang Yang. Towards personalized federated learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[23] Yiqiang Chen, Xin Qin, Jindong Wang, Chaohui Yu, and Wen Gao. Fedhealth: A federated transfer learning framework for wearable healthcare. *IEEE Intelligent Systems*, 35(4):83–93, 2020.

[24] Tao Yu, Eugene Bagdasaryan, and Vitaly Shmatikov. Salvaging federated learning by local adaptation. *arXiv preprint arXiv:2002.04758*, 2020.

[25] Yihan Jiang, Jakub Konečnỳ, Keith Rush, and Sreeram Kannan. Improving federated learning personalization via model agnostic meta learning. *arXiv preprint arXiv:1909.12488*, 2019.

[26] Fabiola Espinoza Castellon, Aurélien Mayoue, Jacques-Henri Sublemontier, and Cédric Gouy-Pailler. Federated learning with incremental clustering for heterogeneous data. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2022.

[27] Hangyu Zhu, Jinjin Xu, Shiqing Liu, and Yaochu Jin. Federated learning on non-iid data: A survey. *Neurocomputing*, 465:371–390, 2021.

[28] Michael Yeung, Evis Sala, Carola-Bibiane Schönlieb, and Leonardo Rundo. Unified focal loss: Generalising dice and cross entropy-based losses to handle class imbalanced medical image segmentation. *Computerized Medical Imaging and Graphics*, 95:102026, 2022.

[29] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[30] Seyed Sadegh Mohseni Salehi, Deniz Erdogmus, and Ali Gholipour. Tversky loss function for image segmentation using 3d fully convolutional deep networks. In *International workshop on machine learning in medical imaging*, pages 379–387. Springer, 2017.

[31] Rahul K Vigneswaran, R Vinayakumar, KP Soman, and Prabaharan Poornachandran. Evaluating shallow and deep neural networks for network intrusion detection systems in cyber security. In *2018 9th International conference on computing, communication and networking technologies (ICCCNT)*, pages 1–6. IEEE, 2018.

[32] Sathyanarayanan Revathi and A Malathi. A detailed analysis on nsl-kdd dataset using various machine learning techniques for intrusion detection. *International Journal of Engineering Research & Technology (IJERT)*, 2(12):1848–1853, 2013.

[33] L Dhanabal and SP Shantharajah. A study on nsl-kdd dataset for intrusion detection system based on classification algorithms. *International journal of advanced research in computer and communication engineering*, 4(6):446–452, 2015.

[34] Zhipeng Li, Zheng Qin, Kai Huang, Xiao Yang, and Shuxiong Ye. Intrusion detection using convolutional neural networks for representation learning. In *International conference on neural information processing*, pages 858–866. Springer, 2017.

[35] Rama Devi Ravipati and Munther Abualkibash. Intrusion detection system classification using different machine learning algorithms on kdd-99 and nsl-kdd datasets-a review paper. *International Journal of Computer Science & Information Technology (IJCSIT) Vol*, 11, 2019.

[36] Nour Moustafa and Jill Slay. Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In *2015 military communications and information systems conference (MilCIS)*, pages 1–6. IEEE, 2015.

[37] Vikash Kumar, Ditipriya Sinha, Ayan Kumar Das, Subhash Chandra Pandey, and Radha Tamal Goswami. An integrated rule based intrusion detection system: analysis on unsw-nb15 data set and the real time online dataset. *Cluster Computing*, 23:1397–1418, 2020.

[38] Ahmed Aleesa, MOHAMMED Younis, Ahmed A Mohammed, and N Sahar. Deep-intrusion detection system with enhanced unsw-nb15 dataset based on deep learning techniques. *Journal of Engineering Science and Technology*, 16(1):711–727, 2021.

[39] Malek Al-Zewairi, Sufyan Almajali, and Arafat Awajan. Experimental evaluation of a multi-layer feed-forward artificial neural network classifier for network intrusion detection system. In *2017 International Conference on New Trends in Computing Sciences (ICTCS)*, pages 167–172. IEEE, 2017.

[40] Aliya Tabassum, Aiman Erbad, Wadha Lebda, Amr Mohamed, and Mohsen Guizani. Fedgan-ids: Privacy-preserving ids using gan and federated learning. *Computer Communications*, 192:299–310, 2022.

[41] Ons Aouedi, Kandaraj Piamrat, Guillaume Muller, and Kamal Singh. Intrusion detection for softwarized networks with semi-supervised federated learning. In

ICC 2022-IEEE International Conference on Communications, pages 5244–5249. IEEE, 2022.

[42] Praneet Singh, Akhil Pankaj, Reshmi Mitra, et al. Edge-detect: Edge-centric network intrusion detection using deep neural network. In *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–6. IEEE, 2021.

[43] Hong-You Chen and Wei-Lun Chao. Fedbe: Making bayesian model ensemble applicable to federated learning. *arXiv preprint arXiv:2009.01974*, 2020.

[44] Naichen Shi, Fan Lai, Raed Al Kontar, and Mosharaf Chowdhury. Fed-ensemble: Improving generalization through model ensembling in federated learning. *arXiv preprint arXiv:2107.10663*, 2021.

[45] Mohammad Rasouli, Tao Sun, and Ram Rajagopal. Fedgan: Federated generative adversarial networks for distributed data. arxiv 2020. *arXiv preprint arXiv:2006.07228*, 2020.