Implementation and Evaluation of Federated Learning for Predicting Car Crashes Using Binary Classification

Lalitha Chandra[†], Prudhvinath[‡]and Sai Tejesh[*]
[†]Department of Data Sciences
University of Houston–Clear Lake, TX, USA
Emails: {eee, ee}@uhcl.edu

***Abstract*—Federated learning (FL) is a machine learning paradigm that trains an algorithm across multiple decentralized edge devices or servers without exchanging local data samples. This approach is particularly advantageous for scenarios where data privacy is paramount or where data cannot be centralized due to regulatory and logistical constraints. FL enables multiple participants to contribute to a comprehensive model, enhancing privacy and reducing data leakage risks as established by prior research. In this paper, we implement federated learning for a binary classification task—predicting car crashes during peak hours using data from three distinct geographical locations. We utilize four laptop devices for the experiment, with one acting as the server and the others as clients. Each device trains a general model on its local data, which are then aggregated on the server using three different techniques: federated averaging, secure aggregation, and robust aggregation. These methods are chosen for their potential benefits in enhancing model performance and security. Federated averaging generally improves learning speed and simplicity, secure aggregation enhances privacy protection, and robust aggregation offers resilience against outliers or skewed data distributions. We evaluate the performance of the aggregated global model using metrics such as the confusion matrix, precision, recall, F-beta, F1, F2, and ROC-AUC. The results demonstrate how each aggregation method affects the efficiency and privacy aspects of federated learning in practical machine learning applications.**

I. Introduction

In recent years, the surge of data across various domains has led to significant advancements in machine learning (ML). Traditional ML architectures often rely on centralized data processing, where data from various sources is collected and stored in a single location for training ML models. While effective for model accuracy, this centralized approach poses several key issues like data privacy, security, and data transmission challenges [1]. Federated learning emerges as a promising solution to this problem by enabling the development of machine learning models across multiple devices or servers without requiring direct access to the data. This approach not only enhances privacy and security but also allows for the utilization of decentralized data, which is particularly beneficial in scenarios where data cannot be centralized due to privacy concerns or data transmission challenges [2].

Federated learning is a decentralized machine learning technique that trains an ML model across multiple devices' data without exchanging them. This means that instead of bringing all data to one central point for training, the model learns from data at its source. The central concept revolves around sending the model to the data, rather than the data to the model. This not only minimizes privacy and security risks but also reduces the bandwidth required to train models over large, centralized datasets. By doing so, federated learning addresses the dual challenges of utilizing rich, diverse datasets for improving model performance while safeguarding sensitive information [3].

Our project applies the principles of federated learning to the problem of predicting car crashes, a task that requires high levels of accuracy and reliability due to its potential implications for safety and traffic management. The use of car crash data from different geographical locations introduces the challenge of learning from decentralized data sources, making federated learning an ideal framework for our objectives. The core idea behind our implementation is to use a binary classification machine learning Neural Network model that predicts whether a car crash will occur (1) or not (0). This prediction is based on various attributes and factors specific to each location's data. Given the sensitivity and potential privacy concerns associated with this data, federated learning offers a way to leverage the information contained within these datasets without compromising individual privacy [4].

To emulate a federated learning environment, we utilized four laptop devices, designating one as the server and the others as client devices. Each client device was assigned data from a specific location, simulating a scenario where each device represents a data source with unique insights into the conditions that may lead to car crashes. The initial step involves sending a general model from the server to the client devices. These devices then train the model locally with their data, ensuring that the raw data remains on the device and is not exposed to the server or other devices. Following the local training phase, the next step involves sending the local trained model weights back to the server by aggregating the learned information in the form of model weights. This aggregation occurs on the server, combining the updates from all client devices to form a global model. This process is crucial as it allows the model to learn from a wide array of data points without the data itself ever leaving the device. To explore the

[*]Corresponding author: Yalong Wu (wuy@uhcl.edu)

effectiveness of different aggregation strategies, we employed three techniques: federated averaging, secure aggregation, and robust aggregation. Each technique offers a different approach by combining the updates and resilience against potential data discrepancies or adversarial attacks.

The process of federated learning involves several key steps, as depicted in Figure 1. Initially, a general model is sent from a central server to various client devices. These clients, each with their unique datasets, train the model locally. After training, they send back only their model's weights to the server. The server then aggregates these weights to update the global model, which encapsulates the learned knowledge from all participating devices without having access to the actual data. This cycle may repeat several times, with the updated global model sent back to the clients for further training, ensuring that the model becomes more accurate and robust with each iteration. Global Model is taken for further analysis on a combined data set from all the clients. Figure 1 illustrates this process, demonstrating the flow from the server to the clients, back to the server during the federated learning cycle, and doing further evaluation on Global model analysis with combined clients' data. Figure 1: Workflow of federated learning showing the distribution of the general model to the devices (Device-1, Device-2, Device-3), local training on clients, the aggregation of model weights on the server to form a Global Model, and the Global Model analysis on combined clients' data sets.

This paper aims to present our findings from implementing federated learning for car crash prediction, focusing on the methodology, the challenges encountered, and the comparative analysis of different aggregation techniques. By doing so, we contribute to the growing body of knowledge on federated learning applications, particularly in scenarios requiring high standards of privacy and data security.

II. Data Model and System Model

We performed data prepossessing techniques on car crash data from three locations downloaded from Data.gov. We standardized formats, corrected errors, and dealt with missing values across all datasets. We cleaned inaccuracies, removed incomplete records, and transformed features like dates and times for analysis. Each client now has clean, consistent data ready for model training.

Server Model Architecture

The foundational stage of our Server Model Architecture involves constructing a baseline neural network using the TensorFlow Keras library. This network is designed to accommodate 13 input variables, a decision reached in consensus with the requirements stemming from our clients' datasets. It incorporates a sequential model structure featuring two hidden layers activated by the ReLU (Rectified Linear Unit) function, chosen for its efficiency in non-linear transformations. The architecture is capped off with an output layer that employs a sigmoid activation function, facilitating the model's ability to produce binary outcomes—specifically, 0 or 1. This design choice is pivotal for enhancing the model's precision in distinguishing between the two classes.

In refining our model, we conducted extensive experiments with various activation functions to ascertain the most effective configuration for our needs. The empirical evidence gathered from these trials indicated that the ReLU function for the hidden layers, combined with the sigmoid function for the output layer, significantly outperforms other candidates in terms of accuracy. This finding is crucial given the binary classification nature of our project, which aims to predict whether a car crash will occur. To quantify the model's performance and guide its training process, we employed the binary cross-entropy loss function. This choice is particularly suited to our binary classification task, as it measures the discrepancy between the true labels and the predicted probabilities, thus providing a clear signal for model improvement. Additionally, the Adam optimizer was selected for its adaptive learning rate capabilities, enhancing the efficiency of the model's learning process. This comprehensive approach, grounded in thorough experimentation and analysis, forms the cornerstone of our project's model architecture, ensuring its robustness and reliability in predicting car crash occurrences.

III. Methods

Our project's methodology was structured to investigate the efficacy of federated learning in predicting car crashes using data from multiple devices. We employed three distinct aggregation techniques to combine local model weights into an aggregated global model. This section outlines the steps taken in data preparation, model training, and evaluation.

Data Preparation:

Data from various devices was collected and preprocessed to ensure consistency and compatibility. This involved cleaning the data to correct any inaccuracies, handling missing values, normalizing numerical inputs, and encoding categorical variables. After preprocessing, we divided the data into training and testing sets, ensuring that the model could be trained and evaluated on different data samples.

Model Training:

We designed a neural network architecture suited to the binary classification task at hand. The network comprised an input layer corresponding to the number of features in our datasets, two hidden layers using the ReLU activation function for non-linear transformations, and an output layer employing a sigmoid function to predict the probability of a car crash occurrence.

Training occurred in two stages:

1. Local Training: Each device trained a local model on its own data and generated weights to be sent to the server.

2. Global Training: After local training, model weights from each device were sent to a central server to be aggregated using one of three techniques—FedAvg, Secure, or Robust aggregation.

Aggregation Techniques:

- FedAvg Aggregation: Here, we computed a weighted average of the local updates, with greater weight given to

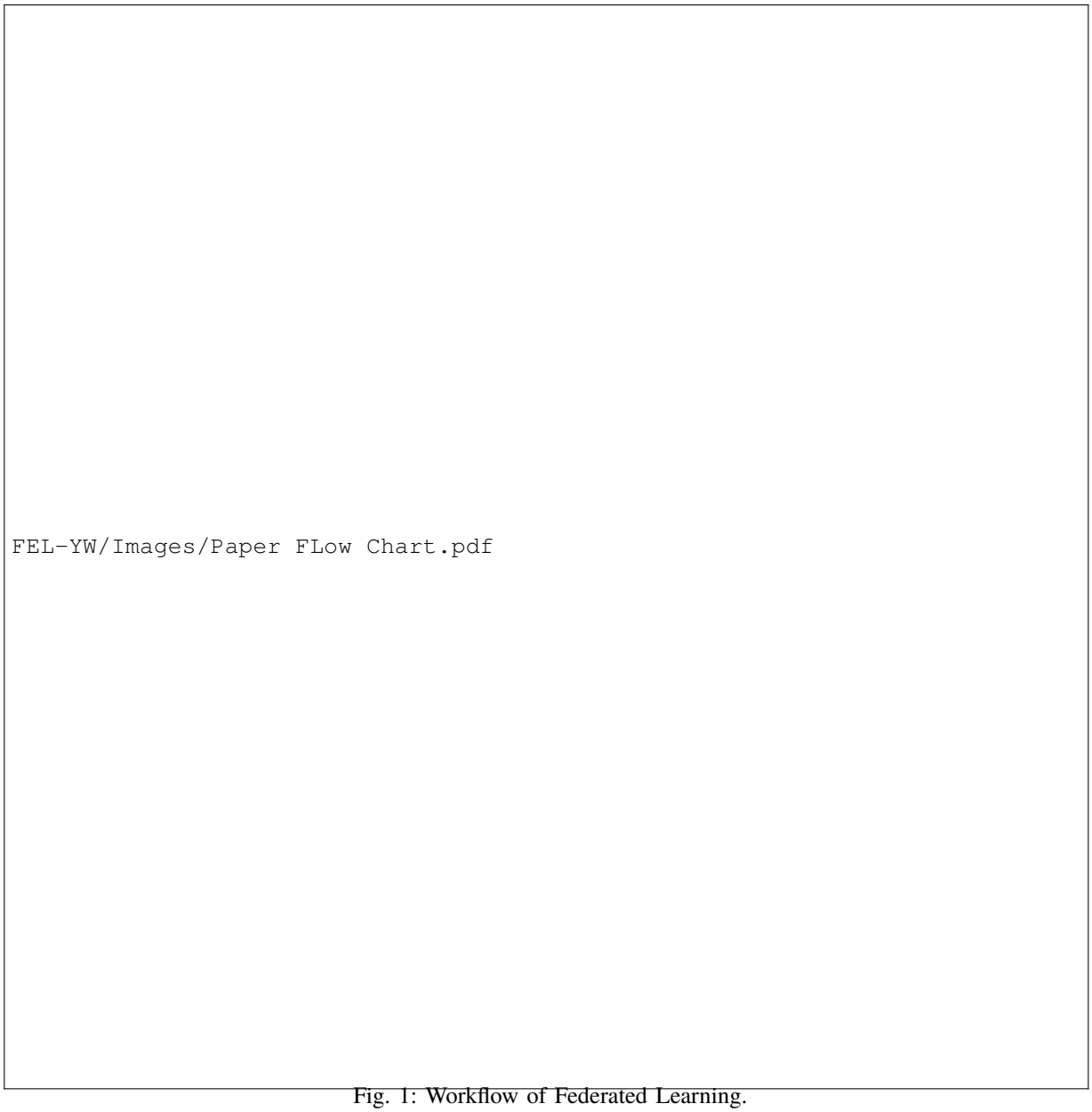

Fig. 1: Workflow of Federated Learning.

devices with more data.

- Secure Aggregation: This technique encrypted the local updates, ensuring that the central server could only decrypt the aggregated update, thus maintaining data privacy.

- Robust Aggregation: We trimmed the most extreme updates from each end of the spectrum to reduce the impact of outliers on the global model.

Evaluation Metrics:

The performance of the locally trained models and the aggregated global model was measured using several metrics: accuracy, recall, precision, F1 and F2 scores, ROC AUC. These metrics provided a comprehensive understanding of each model's performance, balancing the importance of correct predictions against false positives and missed crashes.

IV. Experiment

A. Data Preprocessing

We collected the car crash data for three locations from Data.gov website [5]. Each data set contains a wealth of information that could potentially influence crash occurrence. This includes environmental conditions like weather, temporal factors such as time of day, and human-related variables like traffic density. Our first task was to clean and prepare this data - a process known as preprocessing. This involved standardizing formats, correcting errors, and dealing with missing values. To ensure consistency between clients and general model from server, we applied the same preprocessing steps across all three datasets:

1. Data Cleaning: We examined the datasets for any inaccuracies or inconsistencies and corrected them where possible.

2. Handling Missing Values: Missing data can skew results, so we removed records with incomplete information, depending on the situation.

3. Data Transformation: We transformed certain features into formats better suited for analysis. For instance, dates and times were broken down into components like hour-of-day and day-of-week.

By the end of preprocessing, each client had a clean, consistent set of data ready for model training.

B. Implementation

Feature Selection

The quality of a machine learning model is heavily dependent on the features it learns from. Each feature was evaluated for its potential impact on the likelihood of a crash:

1. Correlation Analysis: Statistical tests were used to find relationships between features and the crash occurrences.

2. Feature Importance Evaluation: Preliminary models were used to rank features by their influence on the predictive outcome.

Features like weather conditions, which could include rain, fog, or snow, and temporal factors, like rush hour versus late-night periods, were among those we considered. We developed new features from the existing ones to capture more complex patterns. For instance, we combined time-of-day and day-of-week to capture weekly traffic flow patterns.

Local Training Process

Local training was carried out on each client device, using their preprocessed datasets. The goal here was to fine-tune our general model to the nuances of each local dataset. We paid close attention to model complexity, avoiding overly complex models that might overfit the local data, thus losing their ability to generalize:

Cross-Validation was used dividing the local data into multiple parts, training the model on some parts, and testing it on others an 80 20 split. It helped ensure that our model's performance was robust across different subsets of the data.

Model Aggregation

After training, client devices sent their updated model weights to the central server for aggregation. This was the moment where the federated aspect of our learning approach came into play with one to many and many to one architecture. Our server did not receive any raw data, only the weights from each local model. Here we used three aggregation techniques:

1. Federated Averaging: Federated averaging is a technique used in distributed machine learning, particularly in federated learning environments. It involves combining the updated weights from multiple models that have been trained locally on separate datasets.

The central server collects the updates from all the clients. It then calculates the average of these updates. If all clients have the same amount of data, this could be a simple mean where each set of updates is given equal weight. However, often some clients have more data than others. In such cases, the server computes a weighted average, where the updates from clients with more data are given more importance.

2. Secure Aggregation: In secure aggregation, the model weight updates undergo encryption prior to transmission to the server. This ensures that each update is securely coded, safeguarding the information against potential attacks. Upon receipt, the server then performs decryption on these encrypted weights, combining them to make the global model. This critical step not only preserves the confidentiality of the data during transit but also strengthens the system against adversarial attacks. By implementing such encryption and decryption mechanisms, secure aggregation significantly enhances the overall security framework, ensuring that the integrity and privacy of the federated learning process are maintained.

3. Robust Aggregation: In robust aggregation, we employ a methodical approach to mitigate the impact of outliers by trimming the extreme values. Specifically, we exclude the top and bottom 5

The resulting global model was a synthesis of all the local learnings, now ready to be evaluated for its predictive power.

Evaluation Metrics

In evaluating the performance of the federated learning model for predicting car crashes, we used a suite of metrics, each offering unique insights into the model's predictive capabilities. The chosen metrics were the confusion matrix, precision, recall, F1 score, F2 score, and ROCAUC. These metrics are particularly suitable for the project because they provide a comprehensive understanding of the model's performance in binary classification tasks, especially in contexts where the balance between false positives and false negatives is crucial.

Confusion Matrix

The confusion matrix is a table used to describe the performance of a classification model on a set of test data for which the true values are known. It displays the number of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). This matrix is fundamental because it lays the groundwork for calculating several other important metrics and gives an immediate visual insight into the model's correct and incorrect predictions.

Precision

Precision, also known as positive predictive value, measures the accuracy of positive predictions. It is calculated as TP / (TP + FP), which means it tells us the proportion of positive identifications that were actually correct. Precision is crucial in scenarios where the cost of a false positive is high. For our project, this means understanding how accurately the model predicts crashes, minimizing false alarms.

Recall

Recall, or sensitivity, measures the model's ability to detect positive samples and is calculated as TP / (TP + FN). It tells us the proportion of actual positives that were correctly identified. In the context of car crash predictions, a high recall is vital because failing to predict a crash could have dire consequences, making it essential that the model captures as many true crash instances as possible.

F1 Score

The F1 score is the harmonic mean of precision and recall.

It provides a single metric that balances both precision and recall, useful when it is tricky to choose one over the other. For predicting car crashes, the F1 score helps ensure that the model is both accurate and sensitive, reflecting its overall effectiveness.

F2 Score

The F2 score weighs recall higher than precision, which is calculated by altering the formula to weight recall more heavily. It is particularly important in situations where missing a positive (i.e., not predicting a crash that does happen) is worse than falsely predicting a positive (predicting a crash that does not happen). Given the critical nature of crash predictions, the F2 score is useful for prioritizing the model's ability to detect all potential crashes, even at the expense of some false alarms.

ROCAUC

The Receiver Operating Characteristic (ROC) curve plots the true positive rate (recall) against the false positive rate (FPR = FP / (FP + TN)), at various threshold settings. The Area Under the ROC Curve (AUC) provides a single measure of the model's performance across all thresholds. A high AUC indicates that the model can effectively distinguish between the classes, making it highly suitable for our project as it signifies the model's ability to discriminate between crash and no-crash instances effectively.

Together, these metrics offer a holistic view of the model's performance, capturing both its accuracy and its ability to handle the uneven costs of false positives versus false negatives. Given the critical importance of accurate and reliable crash predictions, these metrics were selected to ensure a comprehensive evaluation of the model's capabilities, prioritizing safety and reliability.

C. Result Interpretation

In our project, we evaluated a machine learning model using three distinct aggregation techniques: Federated Averaging (FedAvg), Secure Aggregation, and Robust Aggregation. These techniques were applied to combine weights from models trained locally on individual devices (LM) into an aggregated global model (GM). The goal was to predict car crashes accurately while considering the privacy and data security aspects inherent in federated learning.

From the results using FedAvg Aggregation, we saw high performance in locally trained models on Device-1, as reflected by the scores in accuracy, recall, precision, and the F1 and F2 scores. However, once these models were aggregated into the global model, the performance metrics, particularly for Device-1, displayed a noticeable drop. This may indicate that while individual devices perform well on their data, the averaging process may dilute some device-specific patterns critical for prediction.

Secure Aggregation showed a similar trend with perfect metrics for the local models on Device-1 and Device-2, but a decline in the global model's performance, with accuracy, recall, and precision falling to zero. This suggests that while secure aggregation provides strong privacy, it may present challenges in maintaining model performance, potentially due to overemphasis on privacy which could affect the utility of the aggregated model.

Robust Aggregation presented varied outcomes, with the local model on Device-2 maintaining perfect scores across all metrics, but other devices showing weaker performance. Notably, the global model scores showed reduced precision and recall in some cases. Robust Aggregation aims to minimize the impact of outliers, and the variations in performance could suggest that some local models might be overly influenced by outliers and anomalies data.

When we consider the global model trained on the combined dataset from all devices, it achieved very good scores across all metrics regardless of the aggregation technique used. This shows that while each device's data may introduce specific biases to the model, the combined dataset is rich and varied enough to train a highly accurate and reliable global model.

In summary, while individual devices often showed strong performance on local data, the aggregation methods affected the global models in different ways. The FedAvg method suggests a potential compromise of device-specific patterns, Secure Aggregation emphasizes privacy at the potential cost of predictive performance, and Robust Aggregation offers a middle ground. The consistent success of the global model trained on all data highlights the importance of diverse training data for developing robust predictive models in federated learning frameworks.

TABLE I: Model Evaluation Metrics using FedAvg Aggregation
LM: Locally Trained model and GM: Global Model

Device	Model	Accuracy	Recall	Precision	fbeta	f1	f2	ROC AUC
Device-1	LM	0.988	0.978	0.996	0.981	0.987	0.981	0.999
Device-1	GM	0	0	0	0	0	0	0.598
Device-2	LM	1	1	1	1	1	1	1
Device-2	GM	0	0	0	0	0	0	0.590
Device-3	LM	0.666	0.329	0.666	0.366	0.440	0.366	0.677
Device-3	GM	0	0	0	0	0	0	0.644
All Data	GM	0.999	1	0.999	0.999	0.999	0.999	0.999

V. Related Works

The advent of big data has spurred significant advancements in machine learning, pushing the boundaries of traditional data processing architectures. Traditionally, machine learning architectures have relied on centralized data processing, where data

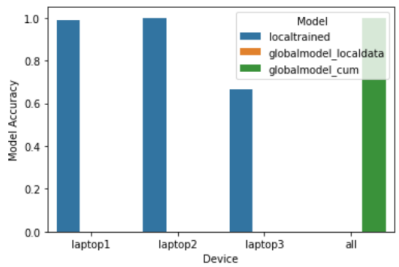

Fig. 2: Comparison of Model Accuracy across Devices Fed Avg

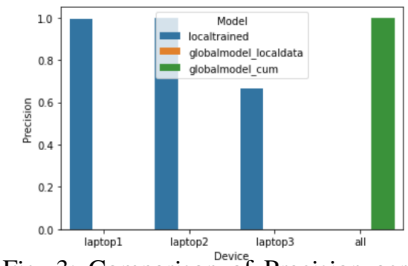

Fig. 3: Comparison of Precision across Devices Fed Avg

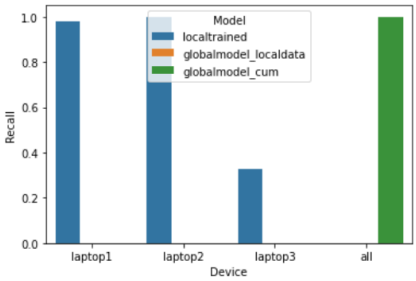

Fig. 4: Comparison of RECALL across

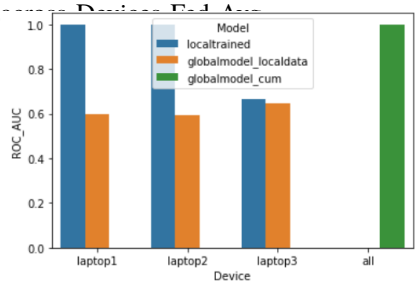

Fig. 5: Comparison of ROC $AUCacrossDevicesFedAvg$

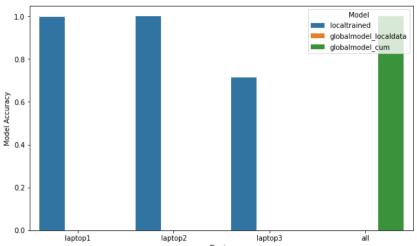

Fig. 6: Comparison of Model Accuracy

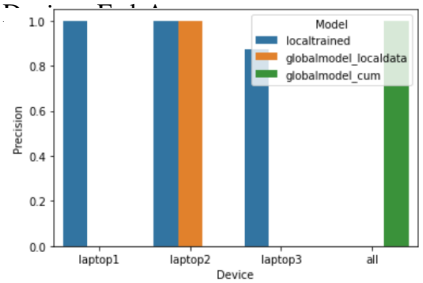

Fig. 7: Comparison of Precision across Devices Secure Agg

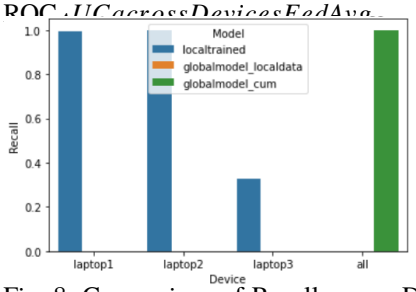

Fig. 8: Comparison of Recall across De-

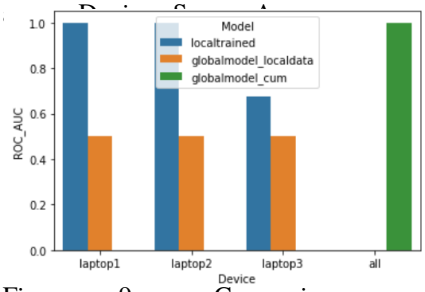

Fig. 9: Comparison of

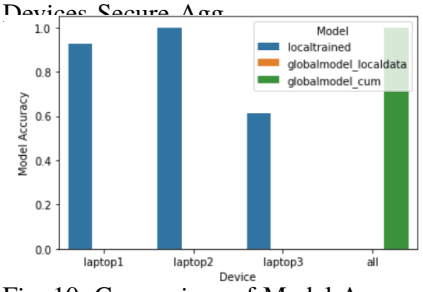

Fig. 10: Comparison of Model Accuracy

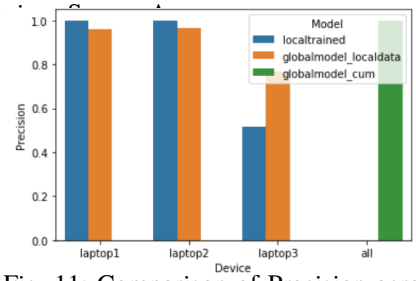

Fig. 11: Comparison of Precision across Devices Robost

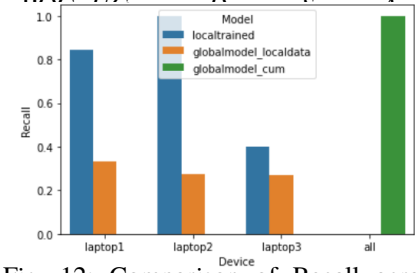

Fig. 12: Comparison of Recall across Devices Robost

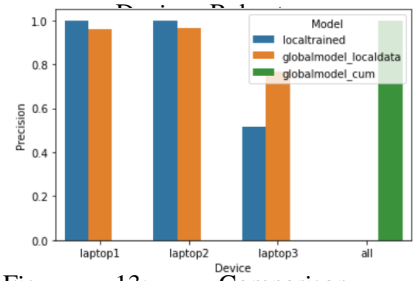

Fig. 13: Comparison of ROC$_{AUCacrossDevicesRobost}$

from diverse sources is aggregated and analyzed in a single location [1]. While this approach enhances model accuracy, it introduces substantial risks concerning data privacy, security, and the logistical challenges associated with data transmission [2].

As a response to these challenges, federated learning has emerged as a transformative approach, enabling the development of machine learning models across multiple decentralized devices or servers without the need to centralize the data [3]. This method not only bolsters data privacy and security but also mitigates the issues related to the massive data transmission of traditional methods. Federated learning is particularly advantageous in scenarios where data cannot be centralized due to privacy laws or practical constraints, making it an ideal framework for sensitive applications like healthcare and personal data services [4].

Federated learning operates on the principle of decentralized machine learning, where a model is trained across numerous devices holding local data, without the data ever leaving its original location. This "model to data" approach significantly reduces privacy and security risks and lessens the bandwidth requirements that are typically necessary for training models over centralized, large datasets [6]. The efficacy of federated learning in utilizing diverse, distributed datasets while safeguarding sensitive information has been documented extensively, showcasing its capacity to maintain high levels of model performance [7].

Our work builds on these foundational concepts by applying federated learning to the task of predicting car crashes using data collected from multiple geographic locations. Given the

TABLE II: Model Evaluation Metrics using Secure Aggregation
LM: Locally Trained model and GM: Global Model

Device	Model	Accuracy	Recall	Precision	fbeta	f1	f2	ROC AUC
Device-1	LM	0.996	0.993	0.998	0.994	0.996	0.994	0.998
Device-1	GM	0	0	0	0	0	0	0.5
Device-2	LM	1	1	1	1	1	1	1
Device-2	GM	0	0	1	0	0	0	0.5
Device-3	LM	0.713	0.329	0.875	0.376	0.478	0.376	0.677
Device-3	GM	0	0	0	0	0	0	0.5
All Data	GM	0.999	0.999	0.999	0.999	0.999	0.999	0.999

TABLE III: Model Evaluation Metrics using Robust Aggregation
LM: Locally Trained model and GM: Global Model

Device	Model	Accuracy	Recall	Precision	fbeta	f1	f2	ROC AUC
Device-1	LM	0.926	0.845	1	0.872	0.916	0.872	0.997
Device-1	GM		0.331	0.959	0.381	0.492	0.381	0.671
Device-2	LM	0.999	0.999	0.999	0.999	0.999	0.999	0.999
Device-2	GM		0.275	0.965	0.321	0.428	0.321	0.653
Device-3	LM	0.610	0.4	0.515	0.418	0.450	0.418	0.696
Device-3	GM		0.270	0.766	0.310	0.4	0.310	0.702
All Data	GM	0.999	1	0.999	0.999	0.999	0.999	0.999

sensitivity and the high accuracy required for such tasks, federated learning offers a strategic advantage. Our approach utilizes a binary classification neural network model to predict potential car crash incidents, a methodology that benefits from federated learning's capability to handle decentralized data sources effectively.

To further explore the effectiveness of federated learning, our project evaluates different aggregation strategies, including federated averaging, secure aggregation, and robust aggregation. Each of these techniques offers unique benefits in terms of model performance and security. Federated averaging is widely recognized for enhancing the speed of convergence in model training [4], secure aggregation focuses on maximizing data privacy [8], and robust aggregation aims to reduce the impact of outliers or skewed data distributions, which is crucial for maintaining the integrity of the model's predictions [9].

VI. Conclusion

In our federated learning project aimed at predicting car crashes, we implemented three aggregation techniques—Federated Averaging (FedAvg), Secure Aggregation, and Robust Aggregation. Each technique demonstrated unique benefits and specific challenges, highlighting the adaptability and potential of federated learning frameworks.

Locally trained models on individual devices showed excellent performance, proving the value of leveraging local data peculiarities. However, the integration of these models into a global model revealed significant performance variations. FedAvg faced challenges in maintaining device-specific accuracy, Secure Aggregation prioritized privacy at the cost of performance, and Robust Aggregation, while protecting against outliers, produced inconsistent results.

Despite these challenges, the global model, trained on aggregated data from all devices, consistently achieved high accuracy and reliability. This underscores the advantage of incorporating diverse data sets, essential in federated learning contexts for developing robust models.

Our project extends the understanding of federated learning, showing that while individual device performance is strong, a combined approach enhances overall model effectiveness. Future work will focus on refining aggregation techniques to optimize the balance between local specificity and global model performance.

References

[1] N. K. Dinsdale, E. Bluemke, V. Sundaresan, M. Jenkinson, S. M. Smith, and A. I. Namburete, "Challenges for machine learning in clinical translation of big data imaging studies," *Neuron*, vol. 110, no. 23, pp. 3866–3881, 2022.

[2] A. Brauneck, L. Schmalhorst, M. M. Kazemi Majdabadi, M. Bakhtiari, U. Völker, J. Baumbach, L. Baumbach, and G. Buchholtz, "Federated machine learning, privacy-enhancing technologies, and data protection laws in medical research: scoping review," *Journal of Medical Internet Research*, vol. 25, p. e41588, 2023.

[3] J. Konečnỳ, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *arXiv preprint arXiv:1610.02527*, 2016.

[4] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[5] Data.gov, "Crash reporting - drivers data," https://catalog.data.gov/dataset/crash-reporting-drivers-data, 2024, accessed: 2024-05-03.

[6] E. Bertino, "Privacy in the era of 5g, iot, big data and machine learning," in *2020 Second IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, 2020, pp. 134–137.

[7] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[8] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1175–1191.

[9] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *International Conference on Machine Learning*. Pmlr, 2018, pp. 5650–5659.