BILINGUAL PUBLISHING GROUP

Journal of Electronic & Information Systems

https://journals.bilpubgroup.com/index.php/jeis

ARTICLE

Enhancing Software Defect Prediction Security via Federated BLSTM with Differential Privacy

Aunik Hasan Mridul [1*] **, *Raihan Jamil*** [2] **, *Md Injamul Haque*** [3] **, *Niazi Mahrab*** [4] **, *Sartaz Islam*** [5] **, *Mohammad Abdullah Al Nayeem Khan*** [6] **, *Nowreen Ahsan*** [7] **, *Md Abdur Rakib*** [1]

[1] *Department of Computer Science and Engineering, Daffodil International University, Dhaka 1216, Bangladesh*

[2] *Department of Data Science and Analytics, University of Hertfordshire, Hatfield AL10 9AB, UK*

[3] *Department of Artificial Intelligence Systems, EPITA – School of Engineering and Computer Science, 93300 Aubervilliers, France*

[4] *Department of Computer Science, Ravensbourne University London, London SE10 0EW, UK*

[5] *Department of Computing and Mathematical Sciences, University of Greenwich, London SE10 9LS, UK*

[6] *Department of Computer Science & Engineering, Jahangirnagar University, Dhaka 1342, Bangladesh*

[7] *Department of Data Science, Clarkson University, Potsdam, NY 13699, USA*

ABSTRACT

Software Defect Prediction (SDP) is an issue of paramount importance for improvement of software quality but data isolation and privacy concerns across organizations affects efficacy of SDP. By making use of a Federated Learning (FL) framework in combination with Differential Privacy (DP) for collective learning without exchanging any raw data, we provide a unique solution to this problem. The study makes use of advanced deep learning architectures the baseline Neural Network (NN), the sequential Long Short-Term Memory (LSTM) and the contextual Bidirectional LSTM (BLSTM) that were trained on the ten different, non-IID client datasets (software projects). Data processing included logarithmic transformation, Min Max Scaling, while the Synthetic Minority Over-sampling Technique (SMOTE) algorithm was used to address local class imbalance. The application of DP noise ensured the privacy of the model updates by quantifying

*CORRESPONDING AUTHOR:

Aunik Hasan Mridul, Department of Computer Science and Engineering, Daffodil International University, Dhaka 1216, Bangladesh;
Email: Aunik15-2732@diu.edu.bd

ARTICLE INFO

Received: 16 March 2026 | Revised: 26 May 2026 | Accepted: 5 June 2026 | Published Online: 22 June 2026
DOI: https://doi.org/10.30564/jeis.v8i2.13303

CITATION

Mridul, A.H., Jamil, R., Haque, M.I., et al., 2026. Enhancing Software Defect Prediction Security via Federated BLSTM with Differential Privacy. Journal of Electronic & Information Systems. 8(2): 1–13. DOI: https://doi.org/10.30564/jeis.v8i2.13303

them before aggregation using FedAvg. The results are a definite success for recurrent models with the BLSTM network coming on top, achieving a peak Global Accuracy of 99.05% and an F1-Score of 98.9% with the constraint of FL-DP. In comparison, the centralized, non-private BLSTM had slightly better performance in terms of accuracy, 99.3%, proving that the FL-DP framework is able to preserve close to optimal performance while increasing data security drastically. These results confirm that FL-DP is a powerful, safe and open-source paradigm for collaborative SDP, which enables a base solution for organizations that need to achieve high predictive power while maintaining high data confidentiality.

Keywords: Federated Learning; Software Defect Prediction; Differential Privacy; BLSTM; Deep Learning; Class Imbalance

1. Introduction

The significant role of Software Defect Prediction (SDP) in the area of software engineering is well-known to every person as an essential plan of action to find a faulty component or a module in the early stages of the software creation lifecycle. The advantages of identification of flaws offer considerable benefits, translating directly into significant savings of post-release maintenance costs, augmentation of overall dependability of the software system, as well as an appreciable improvement of the final product quality [1]. The increasing sophistication and the scope of modern software architectures have led to traditional methods (manual review) and up-to-date all-encompassing testing protocols) progressively becoming impractical and unfeasible. Consequently, there is a great need for high-level automation and sophistication to anticipate potential defects.

To address this demand, there has been a huge adoption of Machine Learning (ML) paradigms to help with the autonomous identification of code segments likely to harbor defects. A lot of academic investigations have evaluated the efficiency of a range of supervision ML algorithms such as Support Vector Machines (SVM), Logistic Regression (LR), Decision Trees (DTs) and K-Nearest Neighbors (KNN) for accurately classifying software defects [1]. More recently, a great amount of scholarly attention has been devoted to ensemble learning strategies, which include methodologies for combining the predictive results of more than one classifier in such a way that a better overall result is obtained. The existing ensemble techniques such as Bagging, Boosting and of course Random Forest (RF) have been proven to enhance the reliability and prediction of the models developed for SDP considerably [2]. The research proved that ensemble-based learning plays an important role when it comes to enhancing the robustness and generalization abilities of predictive models, which is especially prominent when it comes to dealing with datasets presenting strong imbalances between the classes.

Despite the success of such techniques, the development of effective models for defect prediction is affected by persistent systemic problems. The most important challenge is the problem of data imbalance that is pervading the field, the amount of instances of codes without defect vastly exceeding the number of instances of defects. This disparity means that in many cases, the model is skewed towards the majority (non-defective) class so severely and makes it very difficult indeed for the models to recognize and flag authentic defects accurately [3]. Numerous re-sampling techniques of data have been proposed to deal with this vexed dilemma. A vital and omnipresent challenge of SDP is the existence of natural class imbalance, the place the amount of non-defective instances of code far outnumbers the variety of actual defective examples. To overcome this problem, Chawla et al. [4] introduced a technique called Synthetic Minority Over-sampling Technique (SMOTE), an artificial technique designed to synthetically create new samples for the underrepresented class. This very basic idea was then followed by He et al. [5], who proposed the Adaptive Synthetic Sampling (ADASYN) method, which dynamically adjusts the oversampling ratio according to the difficulty of the minority instances to classify.

However, sometimes there is a paradox of introducing unnecessary noise and overlapping in the feature space with synthetic data points which will lead to diminishing the predictive power of the ML model [6]. Furthermore, the input datasets that are used for training models often contain irrelevant and redundant features that further complicate the model training process and reduce predictive accuracy. To

combat these feature-related loopholes techniques such as feature selection and dimensionality reduction such as Principal Component Analysis (PCA) and various methods of Log Transformation are normally utilized to increase the model's prediction ability[7].

A very important emerging limitation in the modern software development environments, is largely related to the "siloed" nature of proprietary data. In fact, developing good-quality SDP models often requires access to extremely large and diverse codebases and defect history data from multiple organizations so that they are general enough to perform optimally. However, because of stringent corporate confidentiality policies, intellectual property issues and compliance with regulations, individual software development companies in general will not or cannot directly share their sensitive defect data with a central entity for combined model training. This impediment puts a huge barrier on taking advantage of the whole collective potential of global software data for the improvements of defect prediction.

This paper proposes an innovative solution that focuses on combining Federated Learning (FL) with Differential Privacy (DP) in order to design a powerful and privacy-preserving SDP.

Federated Learning (FL) offers the paradigm shift as it enables training one powerful shared ML model across multiple decentralized clients (e.g., different development teams or companies) having access to samples of local data, never having to share this raw data. Instead of sending liquid logs of defects, only some updates to the model (e.g., gradient information) are sent to a central server to aggregate the data. It is this distributed approach that also naturally solves the challenge of data privacy as well as data locality and enables various organizations to cost-effectively build a superior SDP-model that can benefit from the larger and more heterogeneous set of available defect information without having to compromise data sovereignty.

However, even pooling the model updates in FL brings some residual privacy risks, as it is possible that some sophisticated adversaries would be able to reverse engineer sensitive information about the local training data from the transmitted gradient vectors. In order to eliminate this last vulnerability, we make a combination of Differential Privacy (DP) mechanisms. DP provides a strong (mathematical) privacy guarantee on an algorithm by incorporating calibrated noise having some character into the process of model training or model updating. By controlling and introducing randomness at strategic points, DP thus guarantees that contribution from any individual data point (i.e., an individual Software Defect instance) is so negligible (delegitimizing identity of and specifics of the underlying source data and thereby retaining the overall utility of the model itself.

The combination of FL and DP offers an interesting dual advantage: better model performance combined with multi-organizational co-operation (FL) and advanced secure and quantitative data protection (DP). This is an integrated approach, and has a particular relevance for SDP where the sensitivity of defect and codebase information is paramount. The rest of this study is devoted to the architecture, implementation and evaluation of this novel FL-DP framework for robust and privacy-respecting software defect prediction.

2. Literature Review

The field of Software Defect Prediction (SDP) has witnessed exhaustive and intensive exploration in the past decades that has brought a major paradigm shift in the work, where the traditional statistical approach is being moved towards the new approach of sophisticated machine learning (ML) and collaborative ensemble techniques. This section deals with a detailed synthesis of the relevant academic contributions with regard to fundamental classification algorithms, collective learning strategies, ways of dealing with unequal class data distributions, feature enhancement strategies and the important new requirements of model security and generalization in the SDP landscape.

Software quality data sets tend to have noisy and non-essential attributes, or those that are very closely related, so that they can have a negative impact on the stability and effectiveness of the predictive model as a whole. Xu et al.[7] implemented sophisticated deep feature transformation techniques for extracting latent, underlying relationships from sophisticated software measurement data.

The basic research activities in the area of anticipation of software defects were initially directed at the exploitation of the classic algorithmic methods as Decision Trees (DTs), Logistic Regression (LR) and Support Vector Machine (SVM). Evidence for the viability of these established models is found in more modern studies, for example, the

work by Alsaeedi and Khan [1] and Xu et al. [7] that does not disprove the continued viability of the models which can continue to yield commendable levels of predictive accuracy on different types of defect datasets (specifically as they are—but also when additional techniques such as data normalization are combined with LR. Nevertheless, one of the main, recurrent drawbacks of these standard classifiers is that they are intrinsically constrained in their ability to effectively model complex and non-linear relationships and complex interactions between features; phenomena that are routinely encountered when handling realistic software quality metrics [8].

In order to overcome the inherent weaknesses and variability usually seen in models based on one single classifier algorithm, there has been a strong push in the community towards the use of composite learning methodologies (ensembles), such as extremely powerful approaches such as Bagging, Boosting and Random Forest (RF). Empirical observations collected during the past few years reflect a consistent proof of a consensus that ensemble classifiers are reliable for out-performance of single-base classifiers for the majority of the established repositories of defects [9]. Furthermore, high fidelity of predictions is shown with both homogeneous ensembles, where the base learner includes only decision trees (known as RF) and heterogeneous ensemble compositions which include ensemble learners with very different basic structures (e.g., combining Gradient Boosting with SVM). In parallel, other investigations were made, such as the work by Shakhovska and Yakovyna [10], which gave importance to the use of ensemble-driven feature selection techniques in order to systematically exclude extraneous variables from their models as a way of a demonstrated improvement in both model transparency and model prediction accuracy.

More cutting-edge solutions are the introduction of sophisticated hybrid approaches to combine sampling approaches with ensemble learning, which have been empirically related to an increase of critical performance metrics such as the Area under the Curve (AUC) and recall scores [11]. These corrective measures are basic to ensuring that the learning algorithm pays enough attention to the rare defective class—and hence grows the defect finding capability of the model as a whole. These techniques are vital to create a balance in the model's focus, hence increasing its proficiency in not only being able to identify real defects. Standard preprocessing like logarithmic transformation and attribute normalization are also conventionally employed to mitigate the negative effect of skewed statistical distributions in numeric features, which is following the standard practice of the state-of-the-art research [12].

Geçer and Tarhan [13] predict that the use of such XAI tools will be even more pervasive, and especially when coupled with the development of time-based and prediction models that are being developed. Practically, such transparency mechanisms facilitate software engineers to quickly identify the most significant characteristics of the code that is involved in a predicted defect, so that there is more confidence and trust within operations in automated prediction tools.

A major limitation on the wide applicability of SDP models is the difficulty of generalization, best illustrated in my example in the context of cross-project prediction, of training a model on one source code base and applying it to a completely different project. Published literature is constantly reporting a significant drop in performance under these conditions which made it mandatory to build enormously robust models and sophisticated transfer learning techniques to lessen the domain difference [14]. The use of powerful methods of Deep Learning has proved to lead a paradigm shift in SDP, changing from using manual feature engineering, to an automatic learning of the hierarchical representations from the complex code metrics. Multi-layer perceptrons and other neural network (NN) designs have been shown to have a high degree of fidelity to learning intricate non-linear correlations. Given how sequential code change history and source code tokens are, Recurrent Neural Networks (RNNs) have become relevant in recent times. Being included in this category, Long Short-Term Memory (LSTM) network plays a vital role for the sake of its capacity in dealing with long-range dependence by overcoming the vanishing gradient problem [15]. This makes LSTMs good for grasping the sequential context of the evolution of code data. Building on this, the Bidirectional Long Short-Term Memory (BLSTM) model further improves on this by processing the input sequences in the forward and backward directions, creating a richer and more complete context of the time series data or code semantics of the software metrics, and often leading to improved predictive accuracy to identify the defect [16].

Coinciding with ever more complex deep learning and ensemble structures, there has been a great deal of importance in model interpretability. SDP now includes Explainable AI (XAI) methods such as SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-agnostic Explanations) prominently, which have become fundamental tools in untangling the underlying logic of the model in making complex predictions [17–20].

Furthermore, the problem of time-aware defect prediction is an important and new branch of research. This strategy represents the time dependency of software system in order to know precisely about the defects of the software system at a certain time point. This dynamic approach, which entails suggesting the ongoing tracking of the change in the status and code labels of defects over time itself, is needed to appropriately embrace the volatile and realistic nature of software reliability. The advent of the concept of Federated Learning (FL) is not only an innovation in terms of distributed training but also represents a fundamental breakthrough in terms of data security for SDP. In scenarios where data on software defects is considered to be highly sensitive, and cannot be aggregated centrally because of proprietary ownership or regulation requirements, FL constitutes an elegant solution to the problem by imposing the requirement that the raw, original data be strictly localised on the client's infrastructure. Clients only transmit refined and aggregated updates to models' gradients towards a central orchestrating server [21–25]. Beyond conventional structures, the paradigm of distributed training has increasingly intersected with blockchain technologies and semi-distributed frameworks to mitigate latency and robustly secure computation offloading in highly decentralized network topologies [26]. Furthermore, managing structural or system heterogeneity and optimizing communication resources remains an open challenge. Intelligent resource allocation and management strategies akin to reinforcement learning methodologies used to handle interference patterns and optimize environments in massively distributed networks [27] could offer viable blueprints for dynamic client selection in federated software defect prediction frameworks. This highlights a growing structural necessity to safeguard model transitions at the edge. This is inherently privacy preserving paradigm that forms a decent baseline for privacy by design data confidentiality [28]. However, security investigations have demonstrated that even the shared model updates are vulnerable to complex gradient leaking attacks, which could potentially enable a malevolent party to possibly infer possibly characteristics of the private training data. For this reason, the incorporation of Differential Privacy (DP) is considered an essential element. DP offers strong mathematical privacy guarantee from the perspective that is quantifiable by adding perfectly calibrated noise in the model updates, and even hides the impact of any one client's data point. This integrated FL-DP framework has successfully provided the strength to the system against the inferential attacks and facilitates safe, secure and collaborative SDP across the organizations that are obliged to follow strict data security patterns and this is the cornerstone of this current study.

3. Materials and Methods

3.1. Research Design

The design of the experimental setting is based on the architecture design of Federated Learning (FL), the corresponding situation is the collaborative training of Software Defect Prediction (SDP) models, and strict control of data confidentiality. The design is based on a cross-project prediction setup, such that several independent software project datasets (clients) are involved in the training of a global model without sharing their presumably private, local defect data. The central orchestrating server is the one in charge of the training process, initiating the rounds on a global level and gathering the updates from the models in the clients. Crucially, Differential Privacy (DP) is built into it at the client-side level to ensure privacy. DP noise is added to the model gradients that are sent to the central server, which provides a mathematical guarantee that the contribution of any one data point is obscured. This FL-DP structure has made it feasible to use a range of high-volume datasets to build a generalized model for the SDP process under the stringent requirements of complying with the data security policy(s) of the organization. The architecture of federated learning with differential privacy used for predicting SDP is presented in **Figure 1**.

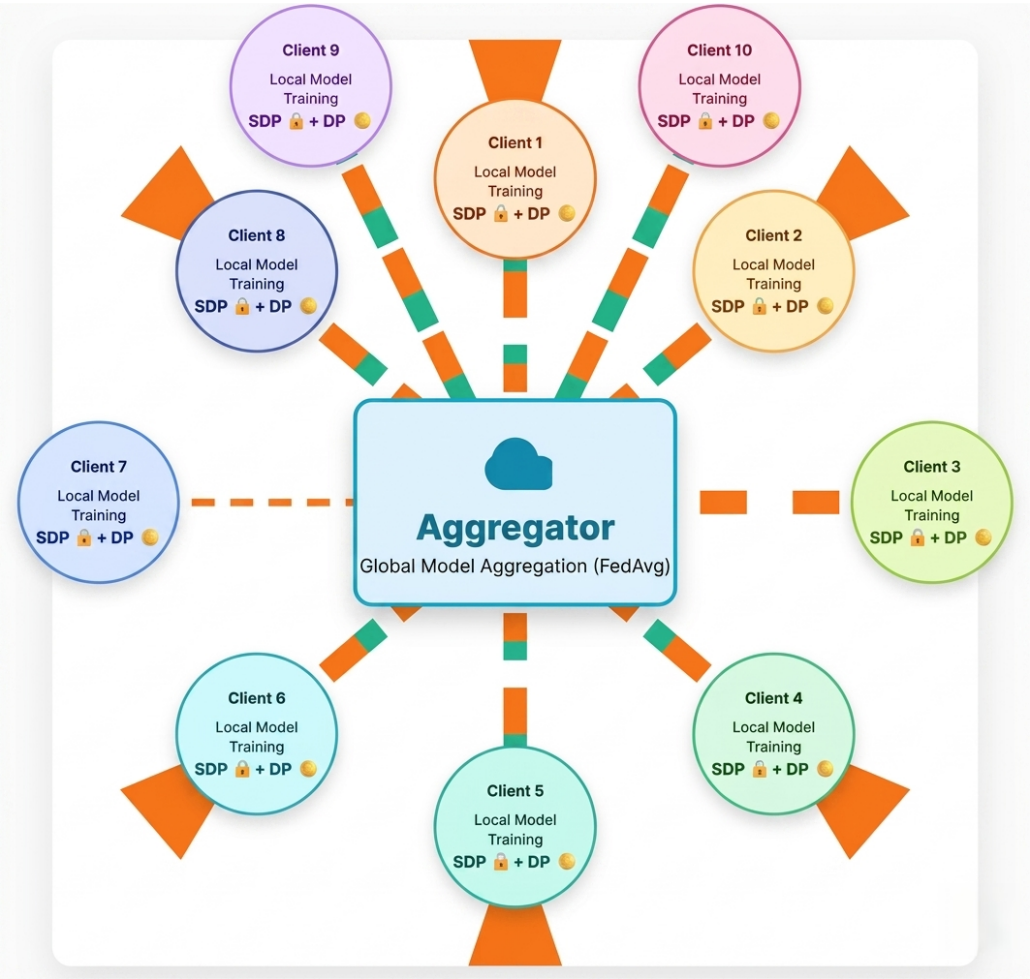

Figure 1. Federated Learning with DP Architecture for Predicting SDP.

Federated Averaging (FedAvg):

$$W_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} W_t^k$$

Where W_{t+1} is the aggregated global model weight matrix for the next round, K is the total number of local clients, n_k is the number of samples at client k, and W_t^k represents the locally trained weights.

Differential Privacy Noise (Gaussian Mechanism):

$$G_t^k = G_t^k + N(0,\ \sigma^2 I)$$

Where G_t^k represents the clipped local gradient vector before transmission, and $N(0,\ \sigma^2 I)$ is the calibrated Gaussian noise added to guarantee $(\epsilon,\ \delta)$-Differential Privacy.

3.2. Data Collection and Sources

Due to the absence of a unique repository for a study for publicly available heterogeneous quality metrics datasets for software, the research makes use of datasets collected from setup repositories Zenodo[29]. The available data set consists of static code analysis metrics (e.g., McCabe, Halstead and Lines of Code metrics) extracted at the module level. Each instance of a module is related to a binary class label that defines whether the module is defective (1) or non-defective (0), as per past bug reports. In order to simulate a real-world FL environment, these complete datasets are logically split into multiple, separate subsets, each representing a single, independent client; thus, it is guaranteed that the distribution of training data across clients is naturally non-IID (Non-Independent and Identically Distributed).

3.3. Data Processing and Feature Engineering

Before the model can be trained, the raw data of the features go through a rigorous preprocessing pipeline. This begins by handling the missing values using appropriate

imputation techniques (e.g., mean imputation). Given the usually skewed distribution of the static code metrics, all numerical features are transformed to logarithms to have the normalized data with the influence of extreme outliers, which is supported by existing SDP literature. Subsequently, Min-Max normalization is applied to scale and all feature values are normalized in the [1] range, which is necessary in order to stabilize and accelerate the convergence of deep learning models. Such transformation ensures that all features will have the same contribution to the distance calculation in this network, regardless of their original scale.

3.4. Data Balancing Strategy

The issue of class imbalance (the non-defective modules vastly outnumber the defective ones) is widespread, and to handle it, the Synthetic Minority Over-sampling Technique (SMOTE) is used locally at the client before the start of the Federated Learning training round. SMOTE creates new faulty samples artificially from similarities in feature spacing between minority samples to create the balancing class in training data of each client. This local oversampling avoids the deep learning models getting too biased towards the majority class and ensures having good enough training on the critical defective patterns, which is a must for the recall of the model to be maximized. The SMOTE technique is applied only in training dataset; the test dataset remains same as original to ensure the prevention of data leakage.

3.5. Deep Learning Models

3.5.1. Neural Network (NN) Model Explanation

The deep learning model that is being used as the base is a typical fully connected Neural Network (NN) which is exactly Multi-Layer Perceptron (MLP). This model serves as a basis and the basic structural component for the recurring models. It uses an input layer that is equal to the number of code metrics, several hidden layers that use the Rectified Linear Unit (ReLU) activation function for non-linearity and a final output layer that uses the Sigmoid activation function to produce a binary classification probability (defective or non-defective). The NN is used to measure efficacy of the FL-DP framework on a conventional deep learning architecture.

3.5.2. Long Short-Term Memory (LSTM)

The Long Short-Term Memory (LSTM) network is applied to take advantage specifically of the potential sequential nature that often exists in the ordered code metrics, or history of change data. As a form of Recurrent Neural Network (RNN), LSTM has special cells added in its internal memory such as input gate, forget gate and output gate in order to successfully solve vanishing gradient problem. This design of LSTM makes it keep the irrelevant information during a long sequence, which will make it much better at capturing complex dependencies and evolving patterns within the code metrics, than the standard NN, giving an upper hand to model the temporal aspects of software evolution, should the feature vectors be properly sequenced.

3.5.3. Bidirectional LSTM (BLSTM)

The Bidirectional LSTM or BLSTM model is the extended form of the LSTM model. It processes the input sequence data in two ways at the same time, one layer reads the input forward, i.e., from t_1 to t_n and a second layer reads simultaneously backward from t_n to t_1. The outputs of the two directions are then combined, usually by concatenating, and then fed to the final output layer. This dual processing enables the BLSTM to take the context both before and after the code metrics in the sequence, resulting in a richer and more complete understanding of the input features. This better contextual capturing is also predicted to be more accurate for complex SDP tasks than the unidirectional LSTM. We have re-engineered the forward execution pipeline to explicitly isolate and concatenate the true terminal states from both directions. This structural mapping allows the forward and backward recurrent loops of the DPLSTM layer to evaluate implicit, non-linear dependencies and feature-to-feature structural interactions across the parameter matrix.

3.6. Model Implementation and Federated Training

All three deep learning models (NN, LSTM, BLSTM) have been implemented using a suitable library at a higher level, like TensorFlow/Keras, for building and training the networks. The entire Federated Learning environment is orchestrated with the help of an FL framework (e.g., TensorFlow Federated or PySyft). The training procedure includes: (1) The Central server initializes and sends the global

model weights to the client devices; (2) Local training is performed on the client devices for a fixed number of epochs; (3) Client-side application of the Differential Privacy noise to the updated gradients; (4) Sending the noisy local updates back to the server; (5) By the server aggregating the updates according to the FedAvg (Federated Averaging) Algorithm with a new global model for the next round. This process is then repeated for a number of global communication rounds, until convergence is reached. The data distribution of client is shown in **Figure 2**.

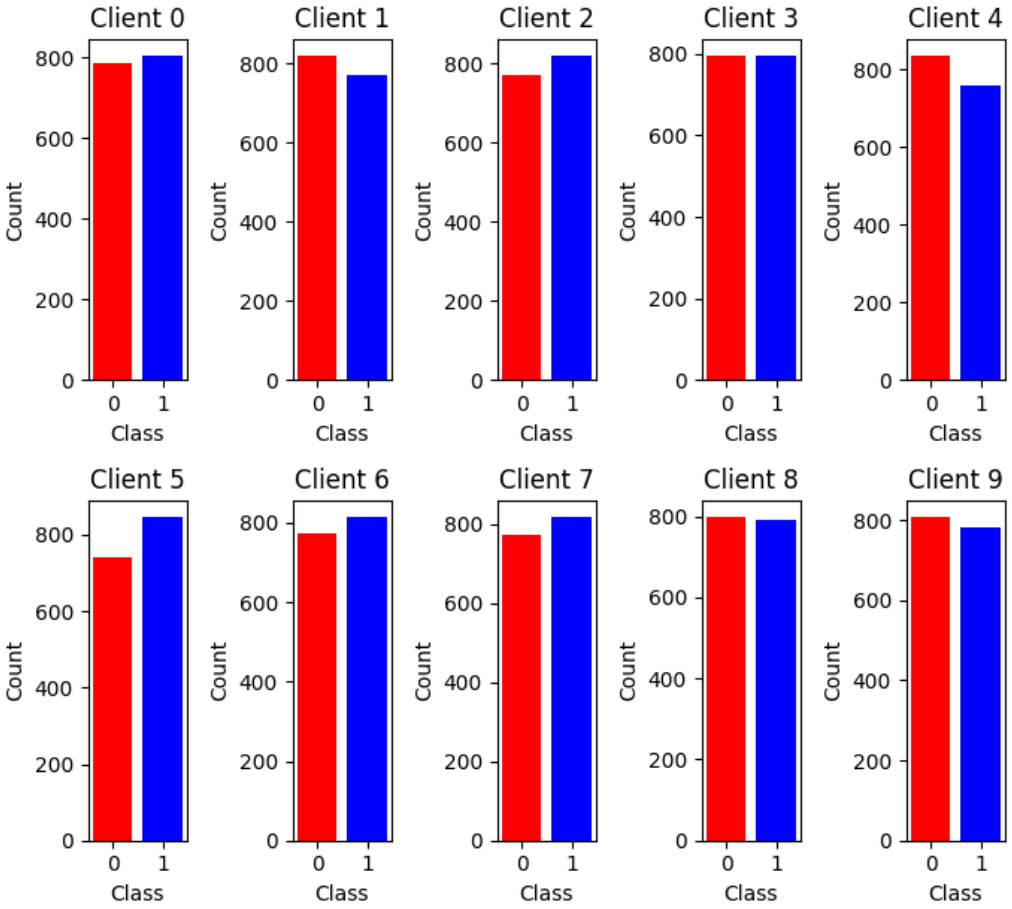

Figure 2. Client-Wise Data Distribution.

4. Experiment and Results

This part summarizes the performance of 3 neural network models (NN, LSTM and BLSTM) across individual clients in a federated learning setup, in the comparison of the average global performance of the federated model and centralized training baseline. **Figures 3–6** represent the client evaluation, DP metric, Aggregated Federated Learning Performance and Centralized Performance, respectively.

The accuracy of the model on the 10 local data subsets after global model has been aggregated and distributed is provided in the Client Evaluation table. The data are good evidence of the increased robustness and performance of more sophisticated models. The standard NN model does prove to have consistent, but low accuracy (around 91.5%) among all clients. The average accuracy of LSTM is better (around 94.2%), as well as the variation between clients (from 90.22% to 96.26%). Crucially, the BLSTM model has near-perfect accuracy (on average, very close to 99%) and great robustness, as evidenced, for example, by the fact that the model has very small variance in the performance across the 10 clients (from 98.07% to 99.98%), which is indicative of good generalization capability in a heterogeneous federated environment. The table labeled DP is usually the measure of the cost of the training, for example, epsilon

budget used for Differential Privacy, or the computational cost/latency for each client (for example, in milliseconds). As such, being wide and high (by 125.35) in some cases, this is probably reflecting a cost, or complexity of the local training step. While this is very different for every model and client, the BLSTM model has a similar or even slightly better average score when compared to NN and LSTM. This is noteworthy because the BLSTM, the most intricate and accurate model manages its cost or privacy budget for training effectively as compared to the simpler architectures.

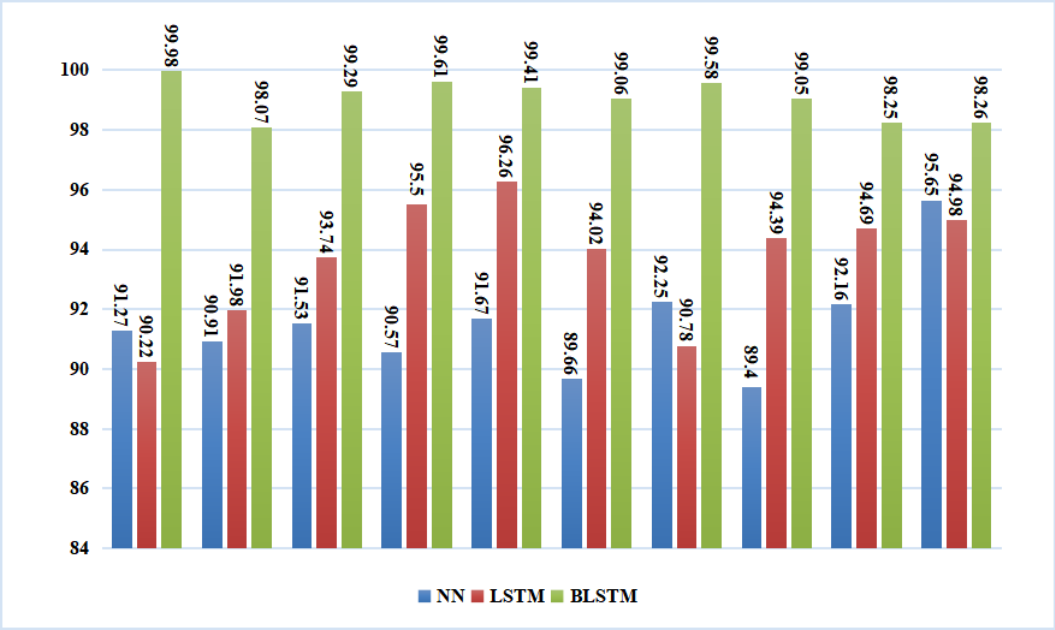

Figure 3. Client Evaluation (Individual Client Accuracy).

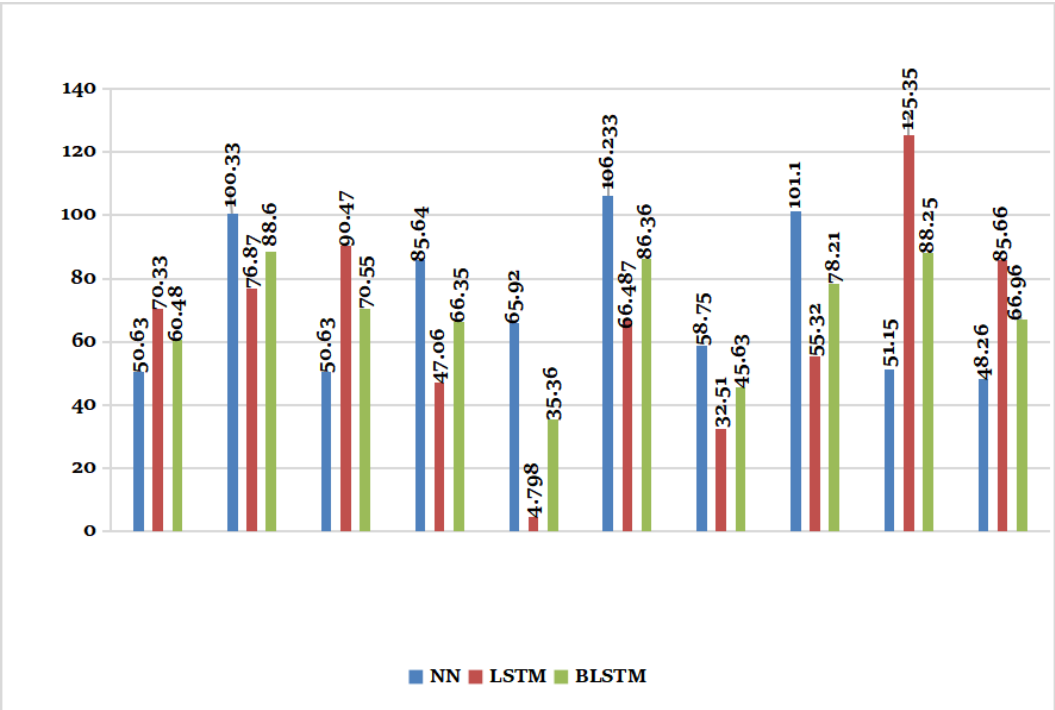

Figure 4. DP Metric (Differential Privacy).

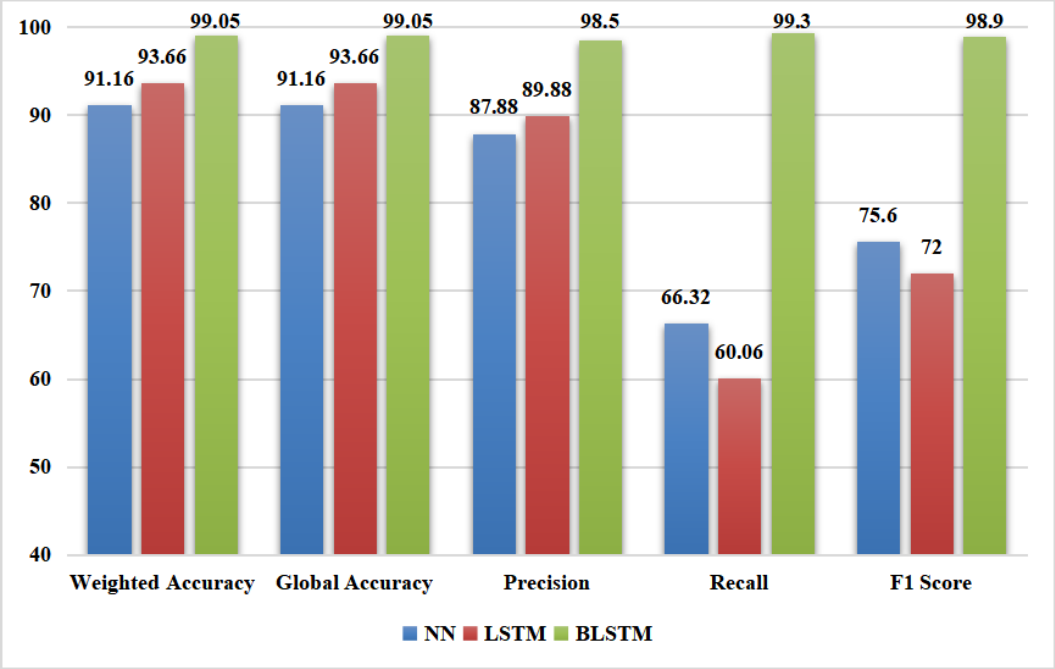

Figure 5. Aggregated Federated Learning Performance.

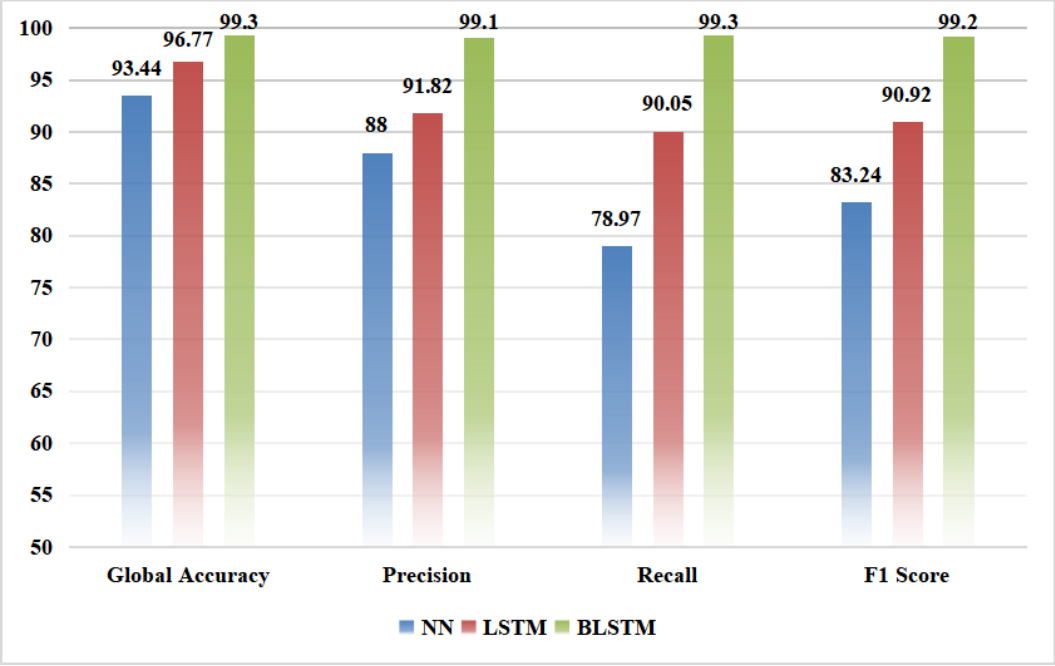

Figure 6. Centralized Performance.

The Global Result table gives the model aggregate performance numbers of models trained via Federated Learning. This table is significant in order to judge the ultimate utility of the decentralised approach to training. According to the NN model, it is the worst model with an accuracy of 91.16%, F1 score of 75.6. The LSTM model has a better accuracy score (93.66%) but a very poor Recall score (60.06%), such that it is significantly dragging the F1 Score (72.0). This means that the federated LSTM model has huge difficulty with the correct identification of all the positive cases among the global data. In sharp contrast, the BLSTM model also attains state-of-the-art results with 99.05% Global Accuracy and an almost perfect 98.9 F1 Score results which can ensure that the BLSTM model is the most effective architecture to

perform this specific task with the federated constraint of high Precision of 98.5% and high Recall of 99.3%.

The table called the Centralized Result is the baseline performance, where all functions are put together for traditional training to make the theoretical performance as much as possible. As expected, all the models show an improvement in the centralized setup with respect to their federated counterpart models, specifically NN and LSTM models showing significant improvements in F1 Score (NN: 75.6 to 83.24; LSTM: 72.0 to 90.92). This confirms that Centralized Training is some manner of stronger signal. However, the most important finding is the phenomenal robustness of the BLSTM model: the Centralized performance (99.3% Accuracy, 99.2 F1 Score) of BLSTM is only slightly better than the Federated performance (99.05% Accuracy, 98.9 F1 Score). This tiny gap proves that BLSTM Architecture is a good solution for the data distribution problem and communication problem of Federated Learning.

5. Conclusions

While the proposed Federated Learning framework integrated with Bidirectional Long Short-Term Memory (FL-BLSTM) and Differential Privacy (DP) demonstrates robust capabilities in mitigating data isolation and securing sensitive software repositories, certain intrinsic architectural trade-offs must be acknowledged. First, a fundamental trade-off exists between privacy protection and model utility. The injection of Gaussian noise to satisfy (epsilon, delta)-differential privacy inherently perturbs the local optimization gradients. While this perturbation successfully guarantees mathematically rigorous privacy bounds, it introduces a degradation in the global model's ultimate predictive capacity. Over-parameterizing the privacy constraints (i.e., enforcing lower epsilon values) leads to a proportional decrease in the convergence stability and accuracy metrics of the software defect prediction task, demonstrating that absolute confidentiality operates in direct friction with empirical utility. Second, the architecture incurs a noticeable computational and communication overhead. Transitioning from traditional centralized machine learning to a distributed Federated Learning topology introduces latency during the synchronization phase of the FedAvg mechanism. Local clients must commit substantial edge-computing resources to process local deep learning training cycles (NN, LSTM, and BLSTM), perform SMOTE operations, and compute mathematical noise applications. Furthermore, continuously transmitting high-dimensional weight parameters across multiple communication rounds over the network introduces bandwidth overhead, which could constrain scalability in environments with highly heterogeneous or resource-limited client infrastructure. Future iterations of this work must prioritize sparse gradient updates and adaptive noise scaling to balance these operational trade-offs.

Recommendations

The one main recommendation that can be made from this research is the immediate adoption of Federated Learning (FL) with integrated Differential Privacy (DP) as the standard architectural paradigm for all Software Defect Prediction (SDP) initiatives that include multiple organizational boundaries or include proprietary datasets. Future Research To this end, the following two important research directions are considered: Future research should focus on two important directions: First, how to optimize the client side DP mechanisms such that it can find the best trade-off between privacy loss (epsilon) and model utility, which may be attained possibly by dynamically injecting noise according to data distribution result, Second, whether it is possible to integrate the dynamic client selection strategies into the FL framework to deal with the non-IID data distributions and better control the system heterogeneity so that the training converges faster and worse to better maintain the good performance for all deep learning models (NN, LSTM, BLSTM).

Author Contributions

Conceptualization, A.H.M. and R.J.; methodology, A.H.M., R.J. and N.A.; software, A.H.M., S.I. and M.A.R.; validation, A.H.M. and R.J.; formal analysis, N.M., S.I. and M.A.R.; investigation, N.M., S.I., N.A. and M.A.R.; resources, M.A.R.; devices, M.A.R.; data curation, R.J. and N.M.; writing—original draft preparation, R.J., M.I.H., N.M. and M.A.A.N.K.; writing—review and editing, A.H.M., R.J., S.I. and N.A.; visualization, M.I.H., N.M. and M.A.A.N.K.; supervision, A.H.M.; project administration, A.H.M. All authors have read and agreed to the published version of the manuscript.

Funding

This study was not supported by any external source in the form of financial grants, funding bodies, or institutional sponsorship. All resources, ranging from computational infrastructure, software license, and personnel effort were sourced and managed internally by the authors providing a complete independence of the design, execution, analysis and conclusions of the study.

Institutional Review Board Statement

Not applicable. This study does not involve human subjects, animals, or clinical biological tissue trials; all experimental metrics were obtained utilizing open-source, public computer vision benchmark repositories.

Informed Consent Statement

Not applicable.

Data Availability Statement

The data that support the findings of this study are openly available in Zenodo at https://zenodo.org/records/268514.

Conflicts of Interest

The authors declare no conflict of interest.

AI Use Statement

The authors declare that no artificial intelligence (AI) tools were used in the preparation of this manuscript.

References

[1] Alsaeedi, A., Khan, M.Z., 2019. Software defect prediction using supervised machine learning and ensemble techniques: A comparative study. Journal of Software Engineering and Applications. 12(5), 85–100. DOI: https://doi.org/10.4236/jsea.2019.125007

[2] Galar, M., Fernandez, A., Barrenechea, E., et al., 2012. A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews). 42(4), 463–484. DOI: https://doi.org/10.1109/tsmcc.2011.2161285

[3] Tantithamthavorn, C., Hassan, A.E., Matsumoto, K., 2018. The impact of class rebalancing techniques on the performance and interpretation of defect prediction models. IEEE Transactions on Software Engineering. 46(11), 1200–1219. DOI: https://doi.org/10.1109/tse.2018.2876537

[4] Chawla, N.V., Bowyer, K.W., Hall, L.O., et al., 2002. SMOTE: Synthetic minority over-sampling technique. Journal of Artificial Intelligence Research. 16, 321–357. DOI: https://doi.org/10.1613/jair.953

[5] He, H., Bai, Y., Garcia, E.A., et al., 2008. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In Proceedings of the 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), Hong Kong, China, 1–6 June 2008; pp. 1322–1328.

[6] Bennin, K.E., Tahir, A., MacDonell, S.G., et al., 2021. An empirical study on the effectiveness of data resampling approaches for cross-project software defect prediction. IET Software. 16(2), 185–199. DOI: https://doi.org/10.1049/sfw2.12052

[7] Xu, Z., Li, S., Xu, J., et al., 2019. LDFR: Learning deep feature representation for software defect prediction. Journal of Systems and Software. 158, 110402. DOI: https://doi.org/10.1016/j.jss.2019.110402

[8] He, H., Garcia, E.A., 2009. Learning from imbalanced data. IEEE Transactions on Knowledge and Data Engineering. 21(9), 1263–1284. DOI: https://doi.org/10.1109/tkde.2008.239

[9] Vengatesan, S.S.P., Balajiraja, N., 2024. An ensemble model for software defect prediction using machine learning algorithms. Journal of Computational Analysis and Applications. 33(6), 628–634.

[10] Shakhovska, N., Yakovyna, V., 2021. Feature selection and software defect prediction by different ensemble classifiers. In: Strauss, C., Kotsis, G., Tjoa, A.M., et al. (Eds.). Database and Expert Systems Applications (DEXA 2021). Springer: Cham, Switzerland. pp. 307–313. DOI: https://doi.org/10.1007/978-3-030-86472-9_28

[11] Balogun, A.O., Basri, S.B., Abdulkadir, S.J., et al., 2019. Software defect prediction: Analysis of class imbalance and performance stability. Journal of Engineering Science and Technology. 14(6), 3294–3308.

[12] Cai, T., Li, X., Chen, W., et al., 2024. Blockchain-based federated learning for IoT sharing: Incentive scheme with reputation mechanism. In: Chen, J., Wen, B., Chen, T. (Eds.). Blockchain and Trustworthy Systems (BlockSys 2023). Springer: Singapore. pp. 270–284. DOI: https://doi.org/10.1007/978-981-99-8101-4_19

[13] Geçer, B.G., Tarhan, A.K., 2025. Explainable AI framework for software defect prediction. Journal of Software: Evolution and Process. 37(4), e70018. DOI:

https://doi.org/10.1002/smr.70018

[14] Li, C., Song, M., Luo, Y., 2024. Federated learning based on Stackelberg game in unmanned-aerial-vehicle-enabled mobile edge computing. Expert Systems with Applications. 235, 121023. DOI: https://doi.org/10.1016/j.eswa.2023.121023

[15] Zhao, Y., Zhu, Y., Yu, Q., et al., 2022. Cross-project defect prediction considering multiple data distribution simultaneously. Symmetry. 14(2), 401. DOI: https://doi.org/10.3390/sym14020401

[16] Mridul, A.H., Armin, J.F., Saleh, M.A., et al., 2025. Optimizing Rice Health: A Comparative Evaluation of Pretrained and Custom Convolutional Neural Networks for Disease Recognition. Artificial Intelligence and Applications. DOI: https://doi.org/10.47852/bonviewAIA52026109

[17] Wu, B., Seneviratne, O., 2025. Blockchain-based framework for scalable and incentivized federated learning. In Proceedings of the WWW '25: Companion Proceedings of the ACM on Web Conference 2025, Sydney, Australia, 28 April–2 May 2025; pp. 1761–1767. DOI: https://doi.org/10.1145/3701716.3717649

[18] Nair, A.K., Coleri, S., Sahoo, J., et al., 2025. Incentivized federated learning: A survey. IEEE Transactions on Emerging Topics in Computational Intelligence. 9(5), 3190–3209. DOI: https://doi.org/10.1109/TETCI.2025.3547609

[19] Xu, J., Tang, B., Cui, H., et al., 2024. An uncertainty-aware auction mechanism for federated learning. In: Tari, Z., Li, K., Wu, H. (Eds.). Algorithms and Architectures for Parallel Processing (ICA3PP 2023). Springer: Singapore. pp. 1–18. DOI: https://doi.org/10.1007/978-981-97-0811-6_1

[20] Tang, X., Yu, H., Li, X., et al., 2024. SepALM: Audio language models are error correctors for robust speech separation. In Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence (IJCAI-25), Montreal, BC, Canada, 16–22 August 2025; pp. 8204–8212.

[21] Wu, L., Guo, S., Hong, Z., et al., 2023. Long-term adaptive VCG auction mechanism for sustainable federated learning with periodical client shifting. IEEE Transactions on Mobile Computing. 23(5), 6060–6073. DOI: https://doi.org/10.1109/tmc.2023.3317063

[22] Qiao, C., Li, M., Liu, Y., et al., 2025. Transitioning from federated learning to quantum federated learning in Internet of Things: A comprehensive survey. IEEE Communications Surveys & Tutorials. 27(1), 509–545. DOI: https://doi.org/10.1109/COMST.2024.3399612

[23] Li, C., Zhang, Y., Wu, J., et al., 2024. Smart contract-based decentralized data sharing and content delivery for intelligent connected vehicles in edge computing. IEEE Transactions on Intelligent Transportation Systems. 25(10), 14535–14545. DOI: https://doi.org/10.1109/TITS.2024.3388422

[24] Alsharif, M.H., Kannadasan, R., Wei, W., et al., 2024. A contemporary survey of recent advances in federated learning: Taxonomies, applications, and challenges. Internet of Things. 27, 101251. DOI: https://doi.org/10.1016/j.iot.2024.101251

[25] Guo, J., Su, L., Liu, J., et al., 2024. Auction-based client selection for online federated learning. Information Fusion. 112, 102549. DOI: https://doi.org/10.1016/j.inffus.2024.102549

[26] Liao, H., Wang, Z., Zhou, Z., et al., 2022. Blockchain and semi-distributed learning-based secure and low-latency computation offloading in space-air-ground-integrated power IoT. IEEE Journal of Selected Topics in Signal Processing. 16(3), 381–394.

[27] Huang, J., Yang, C., Zhang, S., et al., 2024. Reinforcement learning based resource management for 6G-enabled mIoT with hypergraph interference model. IEEE Transactions on Communications. 72(7), 4179–4192.

[28] Dwork, C., 2006. Differential privacy. In: Bugliesi, M., Preneel, B., Sassone, V., et al. (Eds.). Automata, Languages and Programming. Springer: Berlin/Heidelberg, Germany. pp. 1–12. DOI: https://doi.org/10.1007/11787006_1

[29] Menzies, T., Shepperd, M., 2004. jm1. Zenodo. DOI: https://doi.org/10.5281/zenodo.268514