WCRG Global Research
WORLD COMET RESEARCH GROUP

International Journal of Artificial Intelligence & Digital Transformation
Vol. 3, No. 1, 2020
Doi: 10.67228/30713315/IJAIDT-2020PI6H3N
PP. 01-12

Original Article

Federated learning frameworks for privacy-preserving collaborative software development

Dr. Priya Natarajan[1], Dr. Shalini Gupta[2]

[1]*Associate Professor, University of Madras, India.*

[2]*Associate Professor, Panjab University, India.*

Received: 05-04-2020 Revised: 05-27-2020 Accepted: 06-03-2020 Published: 06-05-2020

ABSTRACT

Collaborative software development often involves multiple distributed teams working on shared codebases, raising significant privacy concerns related to proprietary code, intellectual property, and sensitive development data. Traditional centralized methods for collaborative model training risk exposing confidential information. This paper proposes a federated learning framework tailored for privacy-preserving collaborative software development, enabling multiple parties to collaboratively train machine learning models on their local code data without sharing raw data. We integrate privacy-enhancing techniques such as differential privacy and secure aggregation to mitigate information leakage. Experimental results demonstrate that the proposed approach achieves competitive model performance while ensuring strong privacy guarantees, offering a practical solution for privacy-sensitive collaborative development environments. The framework lays the groundwork for secure and efficient cooperation in distributed software engineering workflows.

KEYWORDS

Federated Learning, Privacy-Preserving, Collaborative Software Development, Differential Privacy, Secure Aggregation, Distributed Machine Learning, Software Engineering, Data Privacy.

ISSN: 3071-3315
PP: 01-12.

1. INTRODUCTION

1.1. Background on collaborative software development

Collaborative software development has become a cornerstone of modern software engineering, enabling distributed teams across different geographical locations to work together efficiently on shared projects. This paradigm leverages version control systems, issue tracking, and communication platforms to facilitate real-time collaboration and continuous integration. The rise of open-source communities and enterprise-level distributed development has further accelerated the adoption of collaborative tools. These environments encourage knowledge sharing, parallel development, and faster innovation cycles by allowing multiple developers to contribute simultaneously. However, this collaboration inherently involves handling diverse and sensitive data sources such as proprietary codebases, architectural designs, and user data, which must be managed carefully to prevent unauthorized access and data breaches.

1.2. Privacy challenges in collaborative environments

Despite the benefits of collaborative development, privacy remains a critical concern. As multiple parties contribute to and access shared repositories, the risk of exposing sensitive intellectual property or confidential information increases. Centralized platforms, which collect and store all development data in a single location, pose significant security risks including insider threats, data leaks, and compliance violations with data protection regulations such as GDPR or CCPA. Furthermore, when machine learning techniques are applied to software repositories to automate tasks like bug detection or code recommendation, the underlying data used for training models may reveal proprietary code snippets or development patterns if not properly protected. The challenge, therefore, lies in enabling effective collaboration and leveraging machine learning capabilities without compromising privacy or data ownership rights.

1.3. Overview of federated learning (FL) and its relevance

Federated learning (FL) emerges as a promising solution to these privacy challenges by enabling decentralized model training across multiple clients without requiring raw data to leave local environments. Unlike traditional centralized learning, where data is aggregated on a central server, FL allows each participant, such as a developer or a team, to train models locally on their private data and share only the model updates. These updates are then aggregated on a central server or in a peer-to-peer manner to create a global model that benefits from the collective knowledge of all participants. This paradigm aligns well with collaborative software development settings, where sensitive code and development data remain local, thereby reducing the risk of privacy violations while still facilitating joint machine learning-driven improvements.

1.4. Motivation for using FL in software development collaboration

The motivation for integrating federated learning into collaborative software development stems from the growing need to harness machine learning for enhancing development workflows without compromising the confidentiality of source code and related artifacts. With increasing adoption of AI-

ISSN: 3071-3315
PP: 01-12.

based tools for automated code review, vulnerability detection, and recommendation systems, organizations face the dilemma of leveraging these advancements while preserving proprietary data secrecy. Federated learning offers a balanced approach by enabling collaborative model training that respects privacy constraints. Additionally, FL can help overcome regulatory hurdles by ensuring data does not cross jurisdictional boundaries. The decentralized nature of FL also enhances robustness and fault tolerance, making it suitable for distributed software teams with heterogeneous data and computing resources.

1.5. Research objectives and contributions

This paper aims to design and evaluate a federated learning framework specifically tailored for privacy-preserving collaborative software development. The primary objectives include developing a system architecture that supports decentralized training on distributed code data, integrating robust privacy mechanisms such as differential privacy and secure aggregation, and addressing challenges posed by data heterogeneity and communication overhead. The contributions of this work are threefold: first, providing a novel application of FL in the context of software engineering collaboration; second, demonstrating the efficacy of privacy-preserving techniques in maintaining data confidentiality without sacrificing model accuracy; and third, offering insights into the practical considerations and potential limitations of deploying FL in real-world software development environments.

2. RELATED WORK

2.1. Traditional collaborative software development tools and privacy issues

Traditional tools that support collaborative software development, such as GitHub, GitLab, and Bitbucket, provide centralized platforms where developers can share, review, and merge code. While these platforms facilitate efficient collaboration, they also centralize data storage, creating single points of failure in terms of security. Centralized systems are susceptible to cyberattacks, insider threats, and unauthorized data access, which can lead to significant breaches of sensitive project information. Moreover, the increasing integration of AI-driven features within these tools such as automated code suggestions and bug prediction necessitates access to large volumes of source code, potentially exposing confidential data during model training or inference processes. Although these platforms implement access controls and encryption, they often lack advanced privacy-preserving mechanisms tailored to machine learning on source code data, thereby leaving gaps that could be exploited.

2.2. Existing privacy-preserving techniques in software development

Various privacy-preserving techniques have been proposed to secure software development data, especially in the context of machine learning applications. Approaches like data anonymization and obfuscation have been used to reduce the risk of sensitive information leakage. Homomorphic encryption allows computations on encrypted data but often incurs high computational costs that limit scalability. Secure multiparty computation (SMPC) enables collaborative computation without exposing individual inputs but can be communication-intensive. Differential privacy introduces controlled noise to model updates or outputs to prevent reverse engineering of the training data. While these techniques

ISSN: 3071-3315
PP: 01-12.

offer valuable privacy guarantees, their application in collaborative software development is still limited due to challenges such as maintaining model performance, handling non-IID (independent and identically distributed) data typical in distributed repositories, and integrating with existing development workflows.

2.3. Overview of federated learning frameworks and applications in other domains

Federated learning frameworks like Google's TensorFlow Federated, PySyft, and OpenFL have shown success in domains including healthcare, finance, and mobile applications, where data privacy is paramount. These frameworks facilitate decentralized training by coordinating model updates among clients, implementing privacy-enhancing protocols, and optimizing communication efficiency. In healthcare, FL has enabled collaborative disease diagnosis models without sharing patient data. In finance, it supports fraud detection across institutions without exposing sensitive transactional information. These applications highlight FL's potential for privacy-aware collaboration. However, despite the structural similarities, software development poses unique challenges such as heterogeneous data formats (source code, bug reports, documentation), variable contribution rates among developers, and the need for integration with version control systems, which existing FL frameworks do not explicitly address.

2.4. Gaps in current research regarding FL in software development

While federated learning is gaining traction, its application specifically to software development remains underexplored. Existing literature often focuses on FL for traditional data types like images, text, or tabular data, but little attention has been given to source code, which presents distinct characteristics such as syntax constraints, structural dependencies, and semantic complexity. Moreover, most FL research assumes relatively uniform client participation and data distribution, which contrasts with the diverse and often imbalanced nature of developer contributions. There is also a lack of comprehensive studies that address the integration of privacy-preserving FL techniques into standard software development tools and workflows. Consequently, there is a pressing need to develop FL frameworks that are optimized for software development contexts, addressing domain-specific privacy requirements, data heterogeneity, and practical deployment considerations.

3. FEDERATED LEARNING FUNDAMENTALS

3.1. Concept and architecture of federated learning

Federated learning is an innovative machine learning paradigm designed to enable multiple clients to collaboratively train a shared model while keeping their training data localized and private. Unlike conventional centralized learning where raw data is pooled into a single server for training, FL distributes the learning process across client devices or local servers. Each participant trains the model locally on their private data and transmits only the resulting model updates (such as gradients or weights) to a central coordinating server. This server then aggregates these updates—usually through averaging or more sophisticated techniques to create a global model that benefits from the collective knowledge of all participants. The architecture of federated learning typically comprises three main

ISSN: 3071-3315
PP: 01-12.

components: client devices or nodes that hold local datasets, a central aggregator that coordinates model updates, and a communication network facilitating exchange between clients and the server. This design inherently preserves data privacy by eliminating the need to transfer sensitive raw data, making it highly applicable to environments where data confidentiality is crucial.

3.2. Types of federated learning: horizontal, vertical, and federated transfer learning

Federated learning can be categorized into three main types based on how data is distributed among participants: horizontal, vertical, and federated transfer learning. Horizontal federated learning occurs when clients have datasets with the same feature space but different sample spaces for example, multiple developers or teams working on similar codebases but with unique files or modules. This is the most common type in collaborative software development. Vertical federated learning applies when participants share the same sample space but have different features; for instance, two organizations developing software but collecting different types of data about the same projects or users. Federated transfer learning combines both horizontal and vertical aspects and is used when datasets differ both in features and samples. This type leverages transfer learning techniques to improve model performance when overlap between data is minimal. Understanding these types helps tailor federated learning frameworks to specific collaboration scenarios, optimizing training efficiency and privacy guarantees.

3.3. Privacy mechanisms in FL: differential privacy, secure multiparty computation, homomorphic encryption

Federated learning inherently improves privacy by keeping data local, but further mechanisms are often necessary to mitigate risks such as model inversion attacks or data leakage through model updates. Differential privacy is a prominent technique that introduces calibrated noise to model updates or parameters, providing quantifiable guarantees that individual data points cannot be inferred from the trained model. Secure multiparty computation (SMPC) allows multiple parties to jointly compute a function over their inputs while keeping those inputs private, facilitating secure aggregation of model updates without exposing raw information. Homomorphic encryption enables computations on encrypted data, allowing the central server to aggregate encrypted model updates and perform necessary operations without decrypting them, thus preserving confidentiality throughout the training process. Combining these privacy-enhancing technologies within FL frameworks ensures robust protection of sensitive data, particularly critical in domains like collaborative software development where proprietary code and design information must remain confidential.

4. PROPOSED FEDERATED LEARNING FRAMEWORK FOR COLLABORATIVE SOFTWARE DEVELOPMENT

4.1. System architecture and components

The proposed federated learning framework for collaborative software development is designed to accommodate the unique requirements of distributed teams working with sensitive code repositories. The system architecture consists of multiple developer nodes, each representing a team or individual contributor maintaining local code data. These nodes independently train local machine learning models

ISSN: 3071-3315
PP: 01-12.

on their private repositories or related development artifacts. A central server acts as the orchestrator, coordinating model updates, performing aggregation, and distributing the refined global model back to the clients. Communication channels between nodes and the server are secured using encryption protocols to prevent interception or tampering. Additionally, the framework integrates privacy-preserving modules to ensure data confidentiality throughout the training lifecycle. The modular architecture allows seamless integration with existing version control systems and continuous integration pipelines, enabling effortless deployment within real-world development workflows.

4.2. Data distribution and management across developers or teams

In collaborative software development, data is inherently distributed, with each developer or team possessing unique and often heterogeneous codebases and development-related metadata. This decentralized distribution aligns naturally with the federated learning paradigm. The framework manages this distribution by allowing each node to locally store and preprocess its data, including source code files, commit histories, bug reports, or documentation. Since the datasets vary in size, quality, and structure, the framework incorporates mechanisms to handle imbalanced data volumes and differing feature representations. Data versioning and synchronization with the team's version control systems ensure consistency and enable incremental updates. By preserving data locality, the framework not only addresses privacy concerns but also reduces bandwidth consumption and latency, critical factors in large-scale, geographically dispersed teams.

4.3. Model training process and aggregation strategies

The model training process initiates with the central server distributing an initial global model to all participating developer nodes. Each node trains this model locally using its private dataset for a predetermined number of epochs or until convergence criteria are met. Following local training, nodes transmit their model updates—not raw data—to the server. The server then aggregates these updates using techniques such as Federated Averaging, which computes a weighted average of the local model parameters, considering factors like dataset size and update quality. To enhance robustness, the framework supports advanced aggregation strategies including anomaly detection to exclude malicious or corrupted updates, and adaptive weighting to prioritize nodes with higher data relevance or model performance. Once aggregated, the updated global model is redistributed to clients, and this iterative cycle continues until the model achieves satisfactory performance. This decentralized training approach maintains privacy while leveraging collective intelligence across distributed teams.

4.4. Privacy-preserving mechanisms integrated

To safeguard proprietary code and sensitive development data during federated training, the framework integrates multiple privacy-preserving mechanisms. Differential privacy is employed by injecting noise into local model updates before transmission, which prevents adversaries from reconstructing individual data points while maintaining model utility. Secure aggregation protocols enable the central server to combine encrypted updates from multiple nodes without accessing individual contributions, thus eliminating risks from a potentially untrusted server. Additionally,

ISSN: 3071-3315
PP: 01-12.

communication channels are protected through end-to-end encryption to prevent eavesdropping or data interception. The framework also supports client authentication and access controls to prevent unauthorized participation. These layered privacy defenses ensure that the collaborative machine learning process complies with data protection regulations and organizational security policies, fostering trust among participating developers.

4.5. Handling heterogeneity and non-IID data issues

One of the key challenges in federated learning for collaborative software development is handling data heterogeneity, as each developer's codebase and associated artifacts can differ significantly in distribution, size, and feature representation—a scenario known as non-independent and identically distributed (non-IID) data. This heterogeneity can degrade model convergence and performance if not properly managed. The framework addresses these challenges by employing personalized federated learning techniques, which tailor global models to local data characteristics through fine-tuning or meta-learning approaches. It also incorporates strategies like clustering clients with similar data distributions to perform grouped training, improving update relevance. Adaptive optimization algorithms help balance contribution disparities and reduce bias introduced by dominant clients. Through these methods, the framework maintains model robustness and accuracy despite diverse and unevenly distributed software development data, ensuring effective collaboration across heterogeneous teams.

5. IMPLEMENTATION DETAILS

5.1. Tools, frameworks, and technologies used

The implementation of the federated learning framework for privacy-preserving collaborative software development leverages a combination of state-of-the-art tools and technologies to ensure robustness, scalability, and security. The system utilizes TensorFlow Federated (TFF) and PySyft as foundational frameworks due to their extensive support for federated learning workflows and privacy-preserving techniques. TensorFlow Federated enables efficient orchestration of distributed training and aggregation, while PySyft provides modules for implementing differential privacy and secure multiparty computation. Communication between client nodes and the central server is secured using TLS encryption protocols, ensuring data integrity during transmission. Version control integration is facilitated through Git APIs to synchronize code repositories locally on developer nodes. Additionally, Docker containers are employed to create isolated and reproducible environments across heterogeneous client machines. For privacy mechanisms, libraries supporting homomorphic encryption and differential privacy noise addition are integrated. These combined technologies form a cohesive ecosystem that supports experimental validation and potential real-world deployment of the proposed framework.

5.2. Dataset description (e.g., distributed code repositories or synthetic data)

For experimental validation, the framework is tested on a combination of real-world and synthetic datasets designed to emulate distributed collaborative software development environments. Real-world datasets consist of open-source code repositories from platforms such as GitHub, distributed

ISSN: 3071-3315
PP: 01-12.

across multiple nodes to simulate developer teams with proprietary codebases. These repositories include diverse programming languages, project sizes, and commit histories to reflect typical heterogeneity encountered in practice. To complement real data and control experimental variables, synthetic datasets are generated with customizable parameters such as code complexity, distribution skew, and noise levels. This synthetic data allows rigorous testing of privacy mechanisms and model behavior under different scenarios, including extreme non-IID distributions. Both dataset types include source code files along with metadata like bug reports, commit messages, and developer annotations, enabling multi-modal model training and evaluation. Dataset preprocessing involves tokenization of source code, feature extraction, and normalization to ensure consistency across clients.

5.3. Experimental setup and evaluation metrics

The experimental setup involves a federated learning environment simulating multiple client nodes, each representing a developer or team with local data. The central server orchestrates model distribution and aggregation rounds. Models used for training include neural networks tailored for code analysis tasks such as bug prediction or code summarization. Training is conducted over multiple communication rounds, with local epochs and batch sizes adjusted to mimic realistic computing constraints. Evaluation metrics encompass both model performance and privacy preservation. For performance, metrics such as accuracy, precision, recall, and F1-score measure the model's ability to generalize across heterogeneous data. Privacy evaluation metrics assess the effectiveness of differential privacy by quantifying privacy budgets (ε values) and measuring resistance to membership inference attacks and model inversion attempts. Communication overhead is evaluated in terms of transmitted data volume and latency. Scalability is analyzed by increasing the number of clients and measuring model convergence speed and resource utilization. This comprehensive evaluation framework ensures a balanced assessment of the framework's effectiveness and efficiency.

6. EXPERIMENTAL RESULTS

6.1. Model performance analysis

The federated learning framework demonstrates competitive model performance across various software engineering tasks despite operating in a decentralized and privacy-preserving manner. Results show that models trained via federated learning achieve accuracy and F1-scores comparable to centralized baselines, indicating that distributed training does not significantly compromise predictive capabilities. Performance remains robust across clients with diverse data distributions, validating the framework's ability to handle heterogeneous, non-IID data. Iterative communication rounds progressively refine the global model, and personalized fine-tuning on local nodes further improves local performance. Analysis reveals that advanced aggregation strategies and adaptive weighting enhance convergence speed and reduce bias towards dominant clients. These findings confirm that federated learning can effectively leverage the collective knowledge embedded in distributed code repositories without centralizing sensitive data.

ISSN: 3071-3315
PP: 01-12.

6.2. Privacy evaluation (e.g., privacy leakage, attack resilience)

Privacy evaluations underscore the framework's strength in mitigating data leakage risks inherent in collaborative software development. Incorporation of differential privacy mechanisms ensures that the risk of reconstructing individual code snippets from model updates remains minimal, as demonstrated by low privacy budget parameters and successful defense against membership inference attacks. Secure aggregation protocols effectively prevent the central server from accessing individual client updates, eliminating a major vulnerability point. Furthermore, homomorphic encryption preserves confidentiality during model update aggregation without incurring prohibitive computational costs. Simulated adversarial scenarios reveal that the framework resists common attacks such as model inversion and gradient leakage, maintaining the confidentiality of proprietary code and development patterns. These results illustrate that robust privacy-preserving techniques can coexist with practical performance in federated learning applications.

6.3. Comparison with centralized and other privacy-preserving approaches

When benchmarked against traditional centralized learning approaches, the federated framework exhibits slightly higher communication overhead but offers significant privacy advantages by eliminating the need to centralize raw data. Compared to other privacy-preserving methods such as homomorphic encryption-only or SMPC-only solutions, the proposed integrated approach achieves a balanced trade-off between computational efficiency, communication cost, and privacy guarantees. While purely centralized models may achieve marginally higher accuracy due to unrestricted data access, the federated framework's performance gap is minimal and acceptable in exchange for enhanced security. Furthermore, the federated approach scales more effectively across distributed teams, supporting incremental model updates without costly data transfers. This comprehensive comparison establishes the federated learning framework as a practical and secure alternative to existing methodologies in privacy-sensitive collaborative software development.

6.4. Scalability and communication overhead analysis

Scalability experiments demonstrate that the framework effectively supports an increasing number of developer clients without significant degradation in model convergence or privacy guarantees. Adaptive aggregation strategies and client clustering help mitigate communication bottlenecks by reducing redundant updates and focusing on relevant data subsets. Although communication overhead naturally increases with the number of clients, optimization techniques such as update compression, selective parameter sharing, and asynchronous communication reduce bandwidth consumption and latency. Resource monitoring indicates that client devices with varying computational capabilities can participate without excessive burden, making the framework suitable for heterogeneous environments. These scalability features ensure the framework's viability for real-world deployments involving large, geographically dispersed software development teams.

ISSN: 3071-3315
PP: 01-12.

7. DISCUSSION

7.1. Benefits and limitations of the proposed framework

The proposed federated learning framework offers substantial benefits by enabling privacy-preserving, collaborative machine learning on distributed software development data. It protects proprietary source code and development artifacts by keeping raw data local, thereby reducing privacy risks and regulatory concerns. The framework's ability to handle heterogeneous and non-IID data enhances its applicability in diverse development environments. Moreover, integration with existing tools and robust privacy mechanisms promotes practical adoption. However, limitations include increased communication overhead compared to centralized training, potential challenges in synchronizing updates among highly heterogeneous clients, and the complexity of implementing advanced privacy protocols without impacting model accuracy. Additionally, client dropout or unreliable network connections may affect training stability. Addressing these limitations is essential to improving robustness and user experience.

7.2. Practical challenges in real-world deployment

Deploying federated learning frameworks in real-world collaborative software development introduces several practical challenges. Ensuring seamless integration with varied development environments, version control systems, and continuous integration pipelines requires significant engineering effort. Managing the diverse computational resources and network conditions across client devices adds complexity to orchestration and scheduling. Privacy regulations and organizational policies may impose additional constraints, necessitating customizable privacy settings. Furthermore, motivating developer participation and managing trust among distributed teams are non-technical challenges that influence adoption. Debugging and monitoring federated training processes also require novel tools due to the decentralized nature of computation. Overcoming these challenges is crucial for translating theoretical benefits into practical, scalable solutions.

7.3. Potential improvements and future work

Future work can focus on enhancing the framework's robustness and efficiency by developing more sophisticated aggregation algorithms that account for client reliability and data quality. Incorporating advanced personalization techniques could further tailor models to individual developer needs while maintaining global knowledge sharing. Research into lightweight privacy-preserving methods may reduce computational and communication overhead. Expanding support for multi-modal data, such as combining code with developer communications or issue tracking, could enrich model capabilities. Additionally, building user-friendly interfaces and monitoring tools would facilitate deployment and adoption. Investigating incentive mechanisms to encourage participation and addressing ethical considerations related to federated learning in software development are promising directions. These improvements will help realize the full potential of privacy-preserving collaborative machine learning in software engineering.

ISSN: 3071-3315
PP: 01-12.

8. CONCLUSION

This paper presented a novel federated learning framework tailored for privacy-preserving collaborative software development, addressing the critical need to protect proprietary code and sensitive development data while enabling effective machine learning-driven collaboration. Through a comprehensive exploration of federated learning fundamentals, privacy mechanisms such as differential privacy, secure multiparty computation, and homomorphic encryption were integrated to safeguard data confidentiality across distributed developer nodes. The proposed system architecture supports decentralized model training on heterogeneous, non-IID data typical of software development environments, leveraging adaptive aggregation and personalization strategies to ensure robust model performance comparable to centralized approaches. Experimental results demonstrated that the framework achieves high accuracy in key software engineering tasks while maintaining strong privacy guarantees, effectively resisting common privacy attacks and minimizing data leakage risks. Additionally, the framework exhibits scalable communication efficiency and resilience to client variability, highlighting its practicality for real-world deployment among distributed teams. Despite these advantages, challenges such as communication overhead, client heterogeneity, and deployment complexities remain areas for ongoing research. The implications of this work are significant, offering software development organizations a viable path to harness AI advancements without compromising intellectual property or regulatory compliance. By maintaining data locality and incorporating advanced privacy techniques, federated learning bridges the gap between collaboration and confidentiality, fostering trust and innovation in distributed development workflows. Future directions include refining aggregation methods, enhancing model personalization, integrating multi-modal development data, and addressing practical deployment barriers through improved tooling and incentive mechanisms. This research contributes to the evolving intersection of machine learning and software engineering by demonstrating how privacy-preserving collaborative intelligence can be realized at scale. Ultimately, the proposed federated learning framework empowers software teams to collaborate more securely and effectively, paving the way for more intelligent, privacy-conscious software development ecosystems.

REFERENCES

[1] McMahan, H. B., Moore, E., Ramage, D., Hampson, S., & y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics* (AISTATS).

[2] Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., ... & Seth, K. (2017). Practical secure aggregation for privacy-preserving machine learning. *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*.

[3] Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., & Zhang, L. (2016). Deep learning with differential privacy. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*.

[4] Yang, Q., Liu, Y., Chen, T., & Tong, Y. (2019). Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology*, 10(2), 1-19.

[5] Li, T., Sahu, A. K., Talwalkar, A., & Smith, V. (2020). Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3), 50-60.

[6] Smith, V., Chiang, C. K., Sanjabi, M., & Talwalkar, A. (2017). Federated multi-task learning. *Advances in Neural Information Processing Systems* (NeurIPS).

ISSN: 3071-3315
PP: 01-12.

[7] Shokri, R., & Shmatikov, V. (2015). Privacy-preserving deep learning. *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*.

[8] Geyer, R. C., Klein, T., & Nabi, M. (2017). Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*.

[9] Li, Q., He, B., & Song, D. (2020). Practical privacy-preserving collaborative learning with gradient compression. *Proceedings of the 29th USENIX Security Symposium*.

[10] Hard, A., Rao, K., Mathews, R., Ramaswamy, S., Beaufays, F., Augenstein, S., ... & Ramage, D. (2018). Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604*.

[11] Nasr, M., Shokri, R., & Houmansadr, A. (2019). Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. *Proceedings of the 2019 IEEE Symposium on Security and Privacy*.

[12] Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., ... & Van Overveldt, T. (2019). Towards federated learning at scale: System design. *Proceedings of the 2nd SysML Conference*.