Open-Source AI-based SE Tools: Opportunities and Challenges of Collaborative Software Learning

ZHIHAO LIN, Beihang University, China
WEI MA[*], Nanyang Technological University, Singapore
TAO LIN, Westlake University, China
YAOWEN ZHENG, Nanyang Technological University, Singapore
JINGQUAN GE, Nanyang Technological University, Singapore
JUN WANG, University of Luxembourg, Luxembourg
JACQUES KLEIN, University of Luxembourg, Luxembourg
TEGAWENDE BISSYANDE, University of Luxembourg, Luxembourg
YANG LIU, Nanyang Technological University, Singapore
LI LI[†], Beihang University, China

Large Language Models (LLMs) have become instrumental in advancing software engineering (SE) tasks, showcasing their efficacy in code understanding and beyond. AI code models has demonstrated their value not only in code generating but also in defect detection, enhancing security measures, and improving overall software quality. They are emerging as crucial tools for both software development and maintaining software quality. Like traditional SE tools, open-source collaboration is key in realising the excellent products. However, with AI models, the essential need is in data. The collaboration of these AI-based SE models hinges on maximising the sources of high-quality data. However, data especially of high quality, often holds commercial or sensitive value, making it less accessible for open-source AI-based SE projects. This reality presents a significant barrier to the development and enhancement of AI-based SE tools within the software engineering community. Therefore, researchers need to find solutions for enabling open-source AI-based SE models to tap into resources by different organizations. Addressing this challenge, our position paper investigates one solution to facilitate access to diverse organizational resources for open-source AI models, ensuring privacy and commercial sensitivities are respected. We introduce a governance framework centered on federated learning (FL), designed to foster the joint development and maintenance of open-source AI code models while safeguarding data privacy and security. Additionally, we present guidelines for developers on AI-based SE tool collaboration, covering data requirements, model architecture, updating strategies, and version control. Given the significant influence of data characteristics on federated learning, our research examines the effect of code data heterogeneity on federated learning performance. We consider 6 different scenarios of data distributions and include 4 code models. We also include 4 most common federated learning algorithms. Our experimental findings highlight the potential for employing federated learning in the collaborative

[*] equal contribution with the first author
[†] the corresponding author

Authors' addresses: Zhihao Lin, Beihang University, China; Wei Ma, Nanyang Technological University, Singapore; Tao Lin, Westlake University, China; Yaowen Zheng, Nanyang Technological University, Singapore; Jingquan Ge, Nanyang Technological University, Singapore; Jun Wang, University of Luxembourg, Luxembourg; Jacques Klein, University of Luxembourg, Luxembourg; Tegawende Bissyande, University of Luxembourg, Luxembourg; Yang Liu, Nanyang Technological University, Singapore; Li Li, Beihang University, China.

ACM XXXX-XXXX/2025/6-ART
https://doi.org/10.1145/nnnnnnn.nnnnnnn

development and maintenance of AI-based software engineering models. We also discuss the key issues to be addressed in the co-construction process and future research directions.

CCS Concepts: • **Software and its engineering → Software maintenance tools**; **Open source model**; • **Computing methodologies → Artificial intelligence**; • **Security and privacy → Domain-specific security and privacy architectures**; **Distributed systems security**.

Additional Key Words and Phrases: Data Privacy, Software Engineering, Open-source Code Model, Federated Learning

ACM Reference Format:
Zhihao Lin, Wei Ma, Tao Lin, Yaowen Zheng, Jingquan Ge, Jun Wang, Jacques Klein, Tegawende Bissyande, Yang Liu, and Li Li. 2025. Open-Source AI-based SE Tools: Opportunities and Challenges of Collaborative Software Learning. 1, 1 (June 2025), 25 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Large language models (LLMs) are profoundly transforming various areas of software engineering (SE). LLMs have significantly influenced the methodologies of SE systems [15, 21]. For example, MetaGPT [20] can generate code based on human language descriptions. It simulates the basic process of software development by mimicking the collaboration among team members and integrating the programming capabilities of LLMs. Although the development mode of MetaGPT is relatively simple compared to agile development [3] and lacks measures for security and quality assurance, it has demonstrated the potential of LLMs in the field of software engineering. Researchers have proposed many code models, which can be grouped into two categories according to the model size: pre-trained code models (small) and foundational code models (large). Pre-trained code model series (like CodeBERT [17], CodeGPT [8], and GraphCodeBERT [19]) and foundational code models (like CodeLlama [50], StarCoder [32], and CodeT5+ [62]) have attracted much attention. Researchers have trained expert models based on pre-trained models or foundational code models, which are also emerging, like code repair models [27]. These code models have gradually become basic tools, called AI-based SE tools, in the field of software engineering [69] such as the issue-addressed tool SWE-agent [71].

Although these models excel in automating repetitive tasks and accelerating the development process, significant challenges remain in ensuring the security, explainability, and trustworthiness of the generated code. Furthermore, the development and sharing of AI code models often receive insufficient attention. Three major limitations hinder the development and dissemination of open-source models: *limited access to high-quality code data*, *insufficient community support*, and *inadequate training hardware resources*. ***First***, it is well known that data quality is crucial for the performance of AI models [45]. The open-source code model is mainly developed and published by a single team based on open-source data. Currently, most open-source code models rely on publicly available code datasets. However, the growth rate of high-quality open-source code data has not kept up with the pace at which LLMs capabilities are improving. According to the law of scaling [28], the performance of a model is directly proportional to the amount of high-quality data. Therefore, to develop more powerful code models, there is an urgent need for more high-quality code data. ***Second***, despite their widespread application in many areas of software engineering, such as vulnerability detection [35], they still lack the strong open-source community support typical of traditional software engineering tools. These open-source models also resemble isolated information islands, where individual entities independently complete the training and release of models using open-source data. In an era when LLMs are beginning to reshape the field of software engineering, this pattern needs to be updated. For example, when an individual notices that current vulnerability detection systems underperform with certain vulnerabilities, they retrain the model using additional data, addressing the issue. However, this enhanced model is only available in the new location and has not been assessed for its ability to maintain its original performance levels. Similarly, another individual implemented comparable upgrades for different flaws in the model. Consequently, users need help finding these updated models and may view these unverified improvements with skepticism. ***Third***, the current unipartite participation model leads to low economic efficiency, as different developers or teams conduct redundant training on the same dataset, and the training process of AI models consumes a significant amount of resources, e.g., electricity [8].

Despite the daunting challenges faced in intelligent software engineering, we can still explore solutions. For the first challenge, commercial IT companies or organizations hold high-quality code data and are reluctant to share it, severely limiting industry innovation and development. Based on our collaborative experience with enterprises, IT companies have a strong sense of protection for even a single line of code commentary, let alone the complete code. If a mechanism could be designed to encourage these companies to share their valuable code assets, the development of intelligent software engineering would be significantly propelled. Furthermore, for the second and third challenges, establishing a model-building process that involves multiple parties and integrates different datasets is more efficient and environmentally friendly than training models in isolation.

Therefore, we focus on federated learning [31]—a distributed machine learning framework aimed at enabling multi-party collaboration under the auspices of data privacy protection. Federated learning safeguards data privacy and compliance, and significantly enhances AI model performance through collaborative modeling. There are two related works. FedCSD [4] uses federated learning for detecting code smells while preserving data privacy. ALMITY [73], designed to address the challenges of applying academic models to real-world industrial applications. Both do not discuss how code data distribution affects federated learning and do not consider the potential ability of federated learning to change the community of open-source code models. Our work has deeply researched the impact of different data distributions on federated learning and outlines how we can use federated learning to govern open-source code models. We consider the different code data distribution and include several pre-trained code models and one large foundation model in our experiments. We include 4 most common federated learning strategies for the model aggregation.

In this work, we outline the co-constructed code-model framework and we explore how data distribution among different participants affect the code model performance in this cooperative framework. We demonstrate through experiments that federated learning can achieve performance similar to the centralized training. We also find that federated learning can achieve better results than individual participants training on their own data while ensuring data privacy for each participant.

The experimental code is available [46]. In summary, our study makes the following contributions:

(1) In view of the current problems of code data sharing and code model maintenance, we discussed a framework for collaborative AI-based SE tools building based on federated learning.
(2) We undertake a thorough investigation to assess the influence of data heterogeneity on the outcomes of federated learning models. This comprehensive study aims to understand how heterogeneity in data can affect the model performance and generalization.
(3) Our experimental results strongly support the potential use of federated learning in bringing together various companies to collaborate on the development of intelligent software engineering, thereby promoting the advancement of this field.

Roadmap. The paper is organized by the following way. Section 2 introduces the background. Section 3 demonstrates our motivation and our solution. Section 4 introduces the data distribution strategies, the dataset, the models and the federated learning algorithms we used. Section 5 shows our experimental results. Section 6 state the challenges and opportunities. Section 7 concludes our work.

2 Background

2.1 Large Language Models in SE

The pre-trained models have significantly enhanced task performance in natural language processing, attributed to their excellent generalization capabilities [29, 39, 40]. Researchers have adapted pre-trained transformer models to code data in the software engineering domain [59]. Based on their pre-training strategies and architectural designs, these models are categorized into three types: autoencoding, autoregressive, and sequence-to-sequence (Seq2Seq) models. Autoencoding models leverage the transformer encoder and adopt strategies like masked language modelling (MLM) [53], which hide certain parts of the code and use the surrounding context to predict them. This allows the utilization of future token information for current predictions. For instance, CodeBERT [17] is trained on the CodeSearchNet [24] dataset and features unique data flow input in addition to regular code and natural language inputs, as demonstrated by GraphCodeBERT [19]. Conversely, autoregressive models, exemplified by CodeGPT [8], rely on causal language modelling (CLM) [16] for pre-training, processing data from left to right while maintaining a transformer decoder structure. Seq2Seq models like CodeT5 [62] and CommitBART [36] employ encoder and decoder mechanisms, with the latter focusing on GitHub commit messages. The advent of LLMs such as ChatGPT [1] further stimulated innovation, resulting in specialized coding models like StarCoder [32], CodeLlama [50], and WizardCoder [38]. These models generally utilize transformer decoders, while CodeT5+ [62] retains an encoder-decoder setup. These LLMs excel in program repair, code generation, and summarization tasks. However, while these models have significantly enhanced various software engineering tasks, their application in the real-world scenario and diverse environments remains challenging [12]. One of the key obstacles is ensuring that these models scale effectively across multiple domains while safeguarding sensitive data, which is the most important point for different companies and organizations to join the collaborative community [51]. Before the emergence of LLMs, software engineering primarily applied deep learning models in two ways: fine-tuning them [56] in combination with specific tasks or using them as feature extractors without altering their weights. Post-LLM, the trend shifted toward contextual learning [55], placing these models within broader workflows and integrating domain knowledge for specific scenario applications. To better enable LLMs to solve specific tasks in the future, methods based on efficient fine-tuning of additional weights like LoRA [23] have gained popularity. These methods also include adapter learning [22], prefix, and prompt learning [26]. Recent empirical studies [39, 40, 68] on LLMs in the software engineering domain demonstrate how LLMs can reshape methods in the field. Tools like MetaGPT [20] for automated code generation exhibit the potential for automating software development through sophisticated LLM Agents. Such unit/fuzz testing efforts also employ iterative prompt engineering to generate test code [70, 76, 78] and ensure software system quality [60].

2.2 Federated Learning

Federated learning, first introduced by Google in 2016 [31], is a distributed learning approach to machine learning. It enables multiple devices or servers to train data in a distributed fashion without the need to share data samples. This approach significantly boosts data privacy and security since all data remains on the local devices or servers at the data source, eliminating the need for data migration. Federated learning is particularly suitable for processing data that contains sensitive information [10], such as personal health records or personal identification information, primary concerns for participants in collaborative efforts. It offers a robust solution to this challenge by enabling organizations to collaboratively train and improve these AI-based SE tools without sharing raw data. By decentralizing model training, federated learning not only protects data privacy but also improves the adaptability and performance of LLMs in heterogeneous and distributed environments. This collaborative approach ensures that LLMs, like CodeLlama and StarCoder, can continuously evolve and meet the demands of increasingly complex software systems while preserving the effectiveness of previous SE tasks and addressing privacy concerns.

In contrast to traditional centralized machine learning, which involves uploading all local data to a single server, or classical distributed learning methods that assume local data follows the same distribution, federated learning allows multiple parties to collaboratively develop a robust machine learning model without any data sharing. Federated learning begins with a central server initializing a global model. This model is then distributed to various clients, who train it locally with their own data. These clients then send their model updates back to the server, which aggregates these updates to improve the global model. This cycle repeats until the model performs satisfactorily across all clients and converges to a stable state. This way, it effectively tackles the critical challenges of data privacy, data security, data access rights, and handling heterogeneous data.

Federated learning can be divided into three categories based on data application scenarios: horizontal federated learning [42], vertical federated learning [72], and federated transfer learning [47]. Horizontal federated learning is applied to data with similar feature spaces but different sample spaces. This means that the datasets include the same type of features, such as the user attributes, but the samples are different. For example, the banks in different regions can use horizontal federated learning to share feature information without sharing actual user information, thereby enhancing model performance; vertical federated learning, on the other hand, applies to situations where the datasets have the same sample space but different feature spaces. In this case, each entities involved have data on the same group of users, but they have collected different type of features. For instance, the department store might have the shopping behavior data on a set of customers, while the bank has the financial information on the same set of customers; federated transfer learning is applied to data with different sample spaces and different feature spaces. This approach combines the benefits of federated learning and transfer learning, allowing a model to benefit from knowledge transfer even when the data domains are distinct.

Federated learning also has produced a new form of learning: a fully decentralized learning framework based on blockchain [34]. Blockchain-based Federated Learning (BCFL) integrates blockchain technology and federated learning [64], aiming to solve issues such as privacy, reliability, and scalability that traditional machine learning frameworks cannot fully accommodate. In BCFL, the central server is replaced by a decentralized blockchain peer-to-peer system, allowing federated learning to be deployed securely and efficiently. By taking advantage of the unique properties of the blockchain, such as being tamper-proof, auditable, and decentralized, BCFL can guarantee the integrity and safety of the federated learning process. To enhance the reliability of the global model, BCFL has also introduced an incentive mechanism to reward honest participants and punish dishonest nodes. In the collaborative development environment of open-source AI code models, BCFL is possible to play a significant role. It is entirely decentralized, replacing the central node with a peer-to-peer network [48], ensuring that no superuser can influence the decision-making of the open-source community. Moreover, the blockchain-based federated learning can effectively reward active participants.

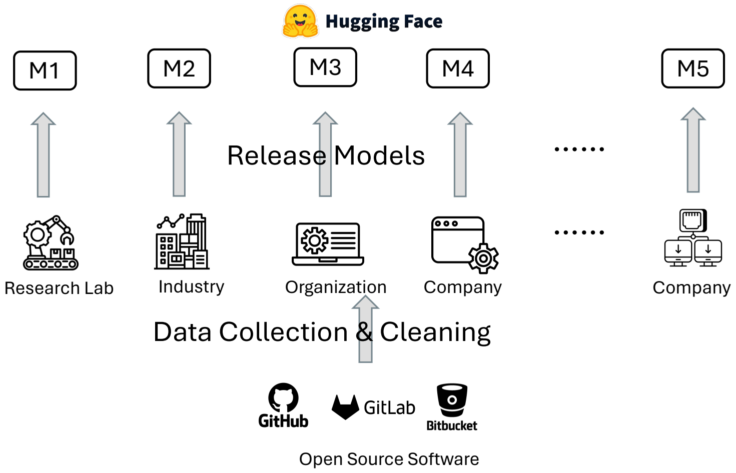

Fig. 1. The Current Governance of Open Source Model

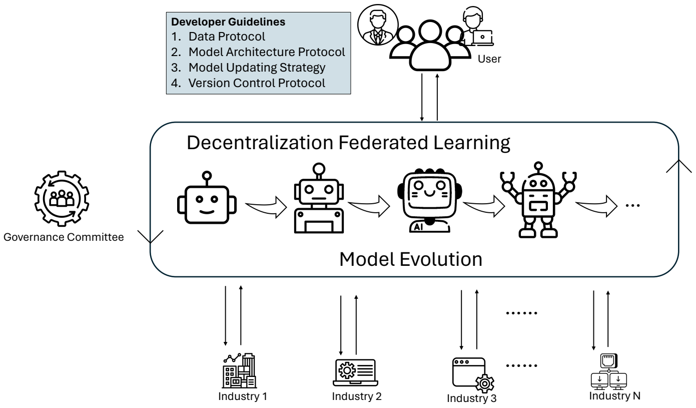

Fig. 2. Decentralization Governance of Open Source Model

3 Proposal

3.1 Motivation

Open-source code and collaboration are key drivers of software engineering development. Without this spirit of open collaboration, the IT achievements we see today, even tech giants like Google, would not exist. With the rise and widespread application of artificial intelligence models in software engineering, they have gradually become the basis for research and development works [5]. However, we note that the management process for open-source code models is facing multiple challenges as we mentioned in Section 1.

The traditional method of open-source model dissemination depends on centralized platforms, such as Huggingface [2], which allows various teams to share the data and models they have collected and trained as shown in Figure 1. However, this way has not truly realized the possibility of team collaboration in developing the models. There may be multiple teams training models on the same dataset, leading to duplicate work. Furthermore, the acquisition of training data for open-source code models mainly relies on platforms like GitHub. However, in the era of big data, the data obtained through these methods has limitations in terms of quality and quantity, and cannot fully capture the complexity of coding practices. Further, specific tasks in software engineering, such as bug fixing [6] and defect detection [44] of different language programming, require high-quality data and labels. However, these scarce resources limit our ability to train high-performance AI code models.

After reflecting on the current way of sharing open-source code models, we draw a question:

> ***Can we co-develop and maintain code models like the development and maintenance methods of open-source software?***

The key to AI code models is high-quality data. Unlike the updates in traditional software code, the release of AI code models involves the weight and architecture of the model. **The biggest challenge** is how to motivate commercial IT organizations to participate in the development of code models and maintain the models while protecting data privacy, which is a key problem we need to solve. Sharing data across organizations may involve sensitive or proprietary code that pertains to the company's core competitiveness.

3.2 Sustainability Concerns in AI Model Development

As AI models grow in complexity, the sustainability of these systems has become an increasingly important concern [49]. LLMs, for instance, require vast amounts of computational power and electricity during the training or fine-tuning, leading to substantial energy consumption. For example, training a state-of-the-art LLM in a specific new domain consume an enormous amount of electricity and requires significant time and GPU resources. The energy and resource consumption not only raise the environmental concerns but also leads to high operational costs [54]. As a result, software engineers and organizations should balance the environmental impacts of AI models with the need for continued innovation as time goes by.

The economic implications of large-scale AI model development present a significant challenge, particularly for the organizations with limited computational resources. The high cost of training and developing an advanced model will be burdensome, with the factors such as high electricity consumption and the expensive hardware, making it difficult to sustain AI systems. These economic factors create an uneven playing field, where smaller companies struggle to compete with the well-funded companies. Therefore, it is crucial to explore strategies that reduce unnecessary costs, making AI development more accessible and equitable.

Previously, numerous related works have focused on the concept of Sustainable AI, proposing various methods to enhance the sustainability of AI systems. For instance, Wu et al. [66] and Van Wynsberghe [58] explored approaches such as model compression and energy-efficient optimization techniques to reduce the environmental impact of large-scale AI systems. They also emphasized the importance of lifecycle carbon footprint monitoring and responsible AI use to assess and mitigate sustainability impacts. Building upon these efforts, and in order to address the aforementioned issues, we propose several potential solutions:

- **Model Compression.** In the decentralized model system, contributors can apply compression techniques locally to reduce the size of updates before transmitting them to the global server. For instance, during the local training phase, participants can use quantization to lower the precision of model parameters, reducing both the storage and communication costs. Similarly, pruning techniques can be employed to eliminate low-impact weights and connections, resulting in a more efficient model without sacrificing performance. These approaches could be evaluated by analyzing the trade-off between compression ratios and the resulting model accuracy, as well as by assessing the overall reductions in energy consumption and communication overhead.

- **Selective Model Update.** Selective updates can be integrated into the system to reduce redundant computations. Participants only update and upload modified parameters that meet a predefined improvement threshold, such as a significant increase in local validation accuracy. This method can be assessed by examining its impact on global model convergence speed, resource utilization, and its ability to maintain robust performance across varying levels of data heterogeneity.
- **Lifecycle Carbon Footprint Monitoring.** The system can integrate a lifecycle carbon footprint monitoring module to dynamically assess the energy consumption of participants during local training and model updates. By tracking these metrics, the governance committee can allocate computational resources more efficiently, prioritizing updates from participants with lower energy consumption. Evaluations could focus on the system's ability to reduce overall energy usage while ensuring that updates from energy-efficient participants contribute meaningfully to global model improvements.

3.3 Decentralization Governance of Open Source Model

Our proposed solution is based on the decentralized federated learning method as shown in Figure 2. Various entities and users collaborate to train a code model using federated learning, and this model has the capability to evolve over time. Federated learning not only provides privacy protection and promotes collaboration but also addresses the sustainability concerns mentioned above. Instead of relying on the vast amounts of open data from the internet, federated learning encourages participants to use their private data to collaboratively maintain an AI model. By decentralizing training processes, federated learning reduces the amount of training data processed by each client and minimizing the reliance on massive centralized data centers [41], which consume significant amounts of energy. This decentralized approach enables organizations to optimize resource consumption, making AI model development more sustainable. Moreover, as organizations increasingly prioritize sustainability, integrating energy-efficient algorithms and ensuring the longevity of AI models becomes essential [9]. Federated learning supports these goals by allowing models to be updated and refined without the excessive energy costs associated with retraining large models from scratch.

To achieve the above target, initially, all participants must reach an agreement on **the developer guideline**. This guideline covers *data protocol*, *model architecture protocol*, *model updating strategies*, and *version control protocol* as explained following:

(1) Data protocol specifies the data formats for training and inference, including quality standards for data and requirements for test data.
(2) The model architecture requires participants to conduct local training based on a predefined exemplary model architecture.
(3) Model updating strategies involve utilizing client weights for model updates, with averaging being a common strategy.
(4) Version control protocol outlines the appropriate checkpoint saving methods, allowing for saves when model performance reaches a certain threshold on a given dataset. It also defines the rules how to control the versions of code models.

As AI technologies evolve, decentralized governance will need to support multi-model collaboration. The role of adaptive systems in facilitating interaction between different models becomes more and more important, allowing for resource sharing and joint problem-solving. Federated learning frameworks must incorporate multi-model collaboration protocols, ensuring that models from different sources can interoperate and adapt autonomously. A Decentralized Autonomous Organization (DAO) could govern this collaboration, allowing participants to vote on model updates, propose improvements, and make decisions in a decentralized and transparent manner.

We also need one governance committee to review the pull request that is submitted by the new participants. This committee is responsible to maintain the developer guidelines and manage the community. Besides, a key challenge in decentralized governance is encouraging participation while ensuring that no single participant gains excessive influence [65]. A token-based contribution system can be an effective approach to reward participants for their active involvement in model development and maintenance. It is designed to foster collaboration as a decentralized mechanism within the federated learning ecosystem. In this system, tokens serve as a unit of value that reflects contributors' efforts and their impact on model improvement. Contributors can earn tokens through measurable contributions, such as submitting effective pull requests or providing high-quality data, and represent the contributor's influence in governance decisions. By enabling contributors with tokens to actively participate in critical decisions, the model ensures that all contributions, regardless of their scale, can directly shape the system's direction while maintaining transparency and fairness in decision-making.

As illustrated in Figure 3, each participant earns tokens based on the number of updates or the improvement in model performance they contribute. These tokens can then be utilized to influence model-related decisions, such as selecting the aggregation strategies, prioritizing specific model updates, or allocating resources for fine-tuning. For example, tokens can be spent to allocate resources such as priority GPU time or enhanced server capabilities for contributors to accelerate their local training or fine-tuning efforts. This mechanism can not only encourage active contributors by ensuring that they can play a larger role in the model governance to shape the direction of the model evolution, but also provide an opportunity to the resource-constrained contributors to access computational resources by contributing high-quality data for training, enabling broader participation and fostering a more inclusive and collaborative ecosystem where contributions of all scales can make some impacts. These tokens would represent individual contributions, which could also be converted into voting power within the DAO, ensuring a fair and transparent decision-making process. To avoid the potential issue of highly active individuals accumulating too much influence through a large number of tokens, which could result in security risks, a "contribution token" creation time could be implemented, similar to the block creation time in blockchain technologies [25]. For example, contributors could be limited to generating tokens at set intervals—such as daily—based on the number of updates and the effectiveness they provide to improve the global model. This approach would govern the distribution of tokens by placing a cap on how frequently they can be earned, encouraging balanced participation while still promoting collaborative efforts. By implementing such a system, more companies and organizations would be encouraged to join the open-source AI model collaboration, reducing duplication of computational resources and energy consumption while increasing overall efficiency.

In this application scenario of our proposed method, we illustrate this with a bug-fixing model as the case study. Initially, one or several entities establish the data protocol and specify the model's architecture. Subsequently, they choose an existing or devise their own federated learning algorithm. A dedicated committee determines the necessity of launching new iterations when managing versions. They begin by preparing a benchmark dataset. All models poised for release must meet a specific performance criterion and the additional dataset to test the improvement. This performance criterion can include different measurements, such as accuracy and robustness. Every version released must delineate the newly incorporated features. At the outset, collaborative training of the model is undertaken, subsequently making it publicly accessible. Later, another team,

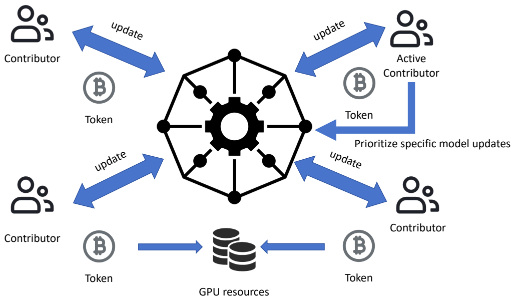

Fig. 3. From Contribution to Influence

endowed with more data, discovered this model. Upon evaluation, they acknowledge its efficacy yet note suboptimal performance for specific code data types. They resolve to refine the model, creating a publicly releasable test dataset in the process. Following the established data protocol, this team retrains the model, uploads the refined model and the new test dataset to the repository dedicated to bug-fixing models, and submits a pull request detailing their actions. The governance committee evaluates this submission and decides on its acceptance. Upon approval, the model is updated using the chosen federated learning algorithm, enhancing its bug-fixing capabilities without compromising existing functionalities.

This approach addresses key challenges: *privacy protection*, *model generalization*, *collaborative innovation*, and *resource optimization*. By training models locally without exchanging data, federated learning inherently protects the privacy of code data. It also facilitates the merging of local datasets from multiple organizations, ensuring a comprehensive and varied data repository that represents a wide array of programming scenarios. The decentralization of model development promotes extensive collaboration. The strategy of distributing model updates rather than data itself efficiently minimizes unnecessary computations, enabling a more strategic allocation of computational resources.

Therefore, we advocate the advantages of federated learning to build a more inclusive, efficient, and privacy-protected new ecosystem for the development of code models in the field of software engineering. This is not only a technological update but also a strategic adjustment towards a more sustainable and collaborative future direction. We expect the new paradigm to create an environment where competitive advantages come from collective progress rather than isolated development, and technological innovation flourishes while respecting data sovereignty. Under the guidance of federated learning, the development of code models can become more dynamic, responsive, and transparent, adapting better to and reflecting the ever-changing coding practices and needs. Therefore, we look forward to a new age of software engineering—a data-driven, collaborative-centered, and privacy-focused intelligent era.

4 Experiment Design

Data is the key driver for the collaborative training and evolution of code models. Model performance is strongly related to the data distribution in federated learning [80]. Therefore, we designed a series of different data distribution strategies and conducted experiments on 4 code models and 5 code tasks to investigate whether federated learning could significantly reduce or improve model performance. We designed 3 research questions as shown in Section 4.3 to study the efficiency and effectiveness of federated learning for AI-based SE tools. These experiments aim to verify the feasibility of federated learning in actual software engineering applications and provide solid empirical support for our framework.

4.1 Data Distribution

In this experiment, we designed 6 different data distribution strategies to evaluate the performance and applicability of federated learning:

Benchmark (Centralization). Serving as the control group, this strategy processes all data centrally, training in a single location. It does not involve distributed processing or storage, adhering instead to the traditional centralized machine learning training approach.

Single Client. We randomly select a subset as the training data for a single client to explore the difference in model performance when trained with data from only a single client versus using federated learning with multiple clients collaborating. The *purpose* of this method is to demonstrate that federated learning can enable each participant to improve model performance while ensuring their own data ownership.

Uniform. Under this strategy, we ensure data is evenly distributed across all participating nodes, with each client possessing approximately the same number of data points and a relatively balanced distribution of labels.

Label Imbalanced. This strategy allows each client to have a similar number of data points but with an uneven distribution of labels within each client. This reflects a common scenario in the real world, where the proportion of data label types varies across different clients, exploring the model's efficiency in handling label-imbalanced data.

Quantity Imbalanced. While maintaining consistent label distribution, there are significant differences in the number of samples between clients in this strategy. It simulates the situation in real environments where some clients may have much more data than others, assessing the model's capability to handle such scale discrepancies.

By Repository. Data is allocated based on its source repository, ensuring that data from the same user or organization belongs to the same client. This method aims to simulate scenarios of data isolation in reality, such as data processing within a company or organization, to evaluate model performance under such data distribution scenarios.

We should notice that **Benchmark (Centralization)** and **Single Client** focus on training models in isolation. Conversely, other strategies are specifically related to the nuances of data distribution within the federated learning framework.

4.2 Datasets and Baseline Models

We utilize the CodeXGLUE dataset [37]. This dataset is a benchmark collection designed to foster machine learning research for program understanding and generation. In our experiment, we choose five tasks: *clone detection*, *defect detection*, *code search*, *code-to-text* and *code completion*. Clone detection and defect detection are crucial for maintaining code quality, while code search and code-to-text involve complex interactions between natural language and code. Code completion focuses on improving developer productivity. These tasks provide an overview of how models perform in real-world programming scenarios, highlighting their abilities in understanding, generating and manipulating code.

For our experiments, we select the following pre-trained models renowned for their effectiveness in code-related tasks: 1) CodeBERT [17] is designed to understand and generate code by leveraging the power of the BERT architecture. It excels in tasks such as code search, code-to-text generation, and more; 2) CodeGPT [8] is fine-tuned specifically for programming languages, making it highly capable of code completion and generation tasks; 3) CodeT5 [63] is an encoder-decoder model designed to understand and generate programming coded; 4) CodeLlama-7b [50] stands as a large language model specifically tailored for coding tasks. We use it into our framework in order to delve deeper into the potential of federated learning for large model fine-tuning in the code domain.

We adopt 4 different, most common federated learning baseline algorithms, FedAvg, FedTrimmedAvg [75], FedMedian and FedProx [33]. FedAvg aggregates the updates from all participating clients by calculating their average. FedTrimmedAvg helps when client data is noisy or has extreme values. It works by ignoring some of the highest and lowest updates from clients and only using the rest. FedProx modifies the local training on each client by adding a penalty term that measures the difference between the local model and the global model. It helps keep each client's update closer to the overall model, especially when their data is very different. FedMedian uses the middle value of client updates instead of the average. This is useful when client data is very different, as it reduces the impact of very high or very low updates.

4.3 Research Questions

We have three research questions as shown in the following:

***RQ1**. Can participation in federated learning enhance model performance while ensuring data privacy?*

We compare model performance between single-client training and multi-client federated learning in code-related tasks. This question aims to assess the efficacy of federated learning in improving model performance over single-client training scenarios for code-related tasks, emphasizing the balance between data privacy and model accuracy.

***RQ2**. In code-related tasks, can federated learning achieve performance comparable to centralized training without disclosing data?*

This research question explores the potential of federated learning to approximate the model performance of centralized training setups, thereby evaluating the feasibility of maintaining high model accuracy in the context of data privacy.

***RQ3**. What impact does data heterogeneity have on the performance of federated learning in code-related tasks?*

This question delves into the effects of data heterogeneity on federated learning outcomes, examining how various approaches to data partitioning affect model performance. It aims to explore the challenges and considerations in managing data diversity across clients in federated learning setups. We define 6 data partitioning strategies as mentioned in Section 4.1 (e.g., uniform distribution, label-imbalanced, quantity-imbalanced) to study this question.

5 Results

We first demonstrate the model performance under federated learning using different data distributions. In the end, we answer our research questions (Section 5.6) based on the experiment results.

5.1 Clone Detection

Settings. We use CodeBERT in this task. We split the dataset into 10 subsets using these distribution strategies: *uniform*, *label-imbalanced*, and *quantity-imbalanced*. The training epochs is 2.

Results. Table 1 displaying the results for the clone detection task reveals insightful patterns regarding the efficacy of federated learning settings compared to traditional centralized training approaches across various datasets. Notably, the F1 score under federated learning are closely aligned with those achieved through centralized training when we use FedAVG as the aggregation strategy, indicating that federated learning is capable of matching the effectiveness of centralized models even when data is distributed across multiple nodes. Furthermore, regardless of the data distribution strategy we use, we find that the models obtained through federated learning significantly outperform those trained on a single client in terms of F1 scores, demonstrating that participating in federated learning can greatly enhance the performance of the obtained models while ensuring data privacy.

The results further illustrate that the federated learning model's performance remains robust even with variations in data distribution. This suggests that federated learning can help maintain model accuracy across different data environments, a critical feature for practical, real-world applications where data is often heterogeneous. The ability to work with imbalanced or distributed data without significant loss of model performance opens new opportunities for federated learning in fields requiring strict data privacy.

Additionally, we can compare the effects of different types of data heterogeneity on the results in this task. We find that when the label distribution is uneven, the performance of the models is much lower than that during uniform distribution, indicating certain limitations of federated learning in dealing with label imbalance. However, in scenarios with uneven quantities, the results obtained by the models are even better than those during uniform distribution, suggesting that heterogeneity in the amount of training data does not adversely affect federated learning and might even enhance the model robustness.

Table 1. The F1 score of Clone Detection

	None-FL	**FedAvg**	**FedTrimmedAvg**	**FedMedian**	**FedProx**
Uniform	-	**0.903**	0.865	0.853	0.849
Label Imbalanced	-	**0.890**	0.860	0.717	0.822
Quantity Imbalanced	-	**0.912**	0.857	0.820	0.854
Centralization	**0.941**	-	-	-	-
Single Client	0.832	-	-	-	-

Note: Bold values in the first three rows denote the best performance among the different federated learning strategies. Bold value in the "Centralization" row indicates the best performance across different approaches.

5.2 Defect Detection

Settings. We use Devign [79] dataset which is collected from two large C programming language open-source projects: QEMU and FFmpeg. And we use accuracy as the evaluation metrics. In this task, we use CodeBERT and CodeT5. We split the dataset into 2 parts in order to simulate the realistic scenarios, each part contains one project. We set training epochs to 5.

Results. Table 2 presents the outcomes for the defect detection task. It is evident that under federated learning conditions, both CodeBERT and CodeT5 achieve results that closely rival those of centralized training across this dataset, surpassing the performance seen with training on a single client. When we use different strategies to aggregate the parameters, we find that the strategies such as FedTrimmedAvg and FedProx assist in obtaining a more refined model.

Moreover, the results suggest that federated learning not only preserves the privacy of each participant's data but also optimizes the resource utilization across different clients. By distributing the computational load and enabling collaboration, federated learning provides a scalable solution to large-scale, multi-organizational software development, where data privacy concerns and hardware limitations are common. These results reinforce the potential of federated learning in scenarios where centralized data collection is not feasible or desirable.

Table 2. The accuracy of Defect Detection

	Aggregation Strategy	CodeBERT	CodeT5
Centralization	-	**62.08**	**62.52**
Single Client	-	57.54	55.53
FL(By Repository)	FedAvg	61.57	58.67
	FedTrimmedAvg	**62.04**	59.96
	FedMedian	61.53	57.65
	FedProx	61.31	**61.05**

Note: Bold values in the "Centralization" row indicate the highest accuracy among the different models. Bold values in the "FL(By Repository)" section represent the best performance for each aggregation strategy.

5.3 Code Search

Settings. In the code search task, we use two models, CodeBERT and CodeT5. Each of these models underwent a triad of training methodologies: centralized, which utilized the complete dataset for training; federated learning, where training was distributed across 10 clients; and single-client training to understand performance under data-constrained conditions. The dataset was split into 10 subsets, with a special emphasis on keeping all repositories from one user within the same subset, to closely mirror practical use cases. We set the training epoch to 2.

Results. In our study on code search as shown in Table 3, CodeBERT and CodeT5 are evaluated across centralized, federated learning, and single-client setups. CodeBERT demonstrates an advantage in federated learning, surpassing its centralized training performance, highlighting its compatibility with distributed data scenarios. Conversely, CodeT5 shows that whatever the aggregation strategies it uses, its performance slightly declined in federated learning compared to centralized training. Both models experienced a drop in effectiveness when trained on data from a single client, indicating the importance of diverse data sources.

The results emphasize the strength of federated learning in handling distributed data without significantly sacrificing model performance. CodeBERT's performance improvements in the federated learning setup suggest that federated learning might even enhance certain aspects of model performance, potentially due to the diversity of data sources involved. This finding highlights federated learning's capacity to create more robust and generalizable models, which are crucial for tasks like code search where data variability is high.

Table 3. The MRR of Code Search

	Aggregation Strategy	CodeBERT	CodeT5
Centralization	-	0.2719	**0.1951**
Single Client	-	0.2520	0.1717
FL(By Repository)	FedAvg	0.2989	**0.1797**
	FedTrimmedAvg	0.268	0.1485
	FedMedian	**0.3022**	0.126
	FedProx	0.2695	0.1478

Note:
- In the CodeBERT, bolded values indicate that FL (specifically FedMedian) outperforms Centralization.
- In the CodeT5, bolded values indicate the highest MRR across all strategies, with FedMedian representing the best FL aggregation strategy.

5.4 Code-to-Text

Settings. In our code-to-test experiment, we evaluated the efficacy of CodeBERT and CodeT5 models under 3 distinct training paradigms: centralized training, federated learning segmented by

Table 4. The BLEU-4 score of Code-to-Text

		Aggregation Strategy	ruby	javascript	go	python	java	php
CodeBERT	Centralization	-	**12.16**	**14.90**	**18.07**	**19.06**	**17.65**	**25.16**
	Single Client	-	7.97	9.99	14.94	15.25	13.86	20.63
	FL(By Repository)	FedAvg	8.90	11.54	16.90	**17.23**	**16.28**	23.80
		FedTrimmedAvg	8.83	11.52	**16.92**	17.20	16.16	23.76
		FedMedian	**8.93**	**11.58**	**16.92**	17.21	16.15	**23.84**
		FedProx	8.75	11.57	16.89	17.19	16.22	23.82
CodeT5	centralized	-	**10.75**	**11.92**	**14.02**	**14.80**	**15.41**	**23.24**
	Single Client	-	9.53	9.73	12.81	11.89	12.44	17.19
	FL(By Repository)	FedAvg	9.95	9.99	13.48	12.00	**12.85**	20.56
		FedTrimmedAvg	9.94	9.95	13.47	11.98	12.82	**20.57**
		FedMedian	9.96	9.97	13.48	11.99	12.81	20.52
		FedProx	**9.99**	**10.00**	**13.50**	**12.03**	12.84	20.60

Note: For both models, bold values in the "Centralization" row represent the highest BLEU-4 scores across different data distributions, while bold values in the "FL(By Repository)" section highlight the best performance for each language and aggregation strategy.

Table 5. The Accuracy of Code-Completion-Token

	Aggregation Strategy	CodeBERT	CodeLlama-7b
Centralization	-	**76.79**	**83.84**
Single Client	-	70.22	81.59
FL(By Repository)	FedAvg	73.45	**83.77**
	FedTrimmedAvg	**73.79**	-
	FedMedian	73.73	-
	FedProx	73.63	-

Note: Bold values in the "Centralization" row represent the highest accuracy for each model, while bold values in the "FL(By Repository)" indicate the best performance for each model and aggregation strategy.

repository, and training on a dataset from a single client. The federated learning approach aimed to mirror real-world data distribution by dividing the dataset into ten subsets, based on repository names, ensuring that all repositories from a single user were grouped together. We match the federated learning rounds to the centralized learning epochs, setting both to ten.

Results. Table 4 shows that centralized training consistently yields the highest BLEU-4 score for both models across all languages, underscoring the benefits of extensive data diversity and volume. Federated learning, while designed to maintain data privacy, sees a moderate decline in BLEU-4 score, highlighting the challenges of aggregating decentralized learning effectively. Nonetheless, federated learning outperforms the single-client scenario, indicating that collaborative learning is preferable to isolated data training, reinforcing the notion that data diversity is crucial for model efficacy. Besides, the impact of different aggregation strategies on the outcomes is minimal in this task.

This result suggests that federated learning is effective for code-to-text tasks, particularly in preserving data privacy across distributed sources without significantly compromising the model's performance. However, a performance gap still exists between federated learning and centralized approaches. Some factors which could help reduce the impact of data heterogeneity and improve the performance of federated learning, these strategies may help narrow the performance gap between federated learning and centralized approaches, particularly in achieving optimal performance in distributed environments.

5.5 Code Completion

Settings. In the code completion task, our experimental setup aimed to evaluate the performance of CodeBERT and the large language model, CodeLlama-7b, under various training paradigms: centralized training, federated learning with dataset division by repository to ensure data privacy, and single-client training to simulate data-constrained environments. We use LoRA to finetune the CodeLlama-7b. We set the training epoch to 5.

Results. Table 5 shows that the centralized training, which is an ideal scenario with access of the entire datasets, yielded the highest accuracy for both models. It is encouraging to notice that the slight decrease in accuracy from centralized to federated learning for both models underscores the trade-offs involved in preserving data privacy through distributed training. Moreover, the further decrease in accuracy from federated learning to single client shows the benefits of collaborative learning in federated learning. Remarkably, CodeLlama-7b outperformed CodeBERT in all tested scenarios. Notably, in the federated learning setup, CodeLlama-7b's performance closely approximated its centralized training accuracy, emphasizing the broad capabilities of federated learning in effectively fine-tuning LLMs.

Moreover, the results suggest that federated learning not only preserves the privacy of each participant's data but also optimizes the resource utilization across different clients. By distributing the computational load and enabling collaboration, federated learning provides a scalable solution to large-scale, multi-organizational software development, where data privacy concerns and hardware limitations are common. These results reinforce the potential of federated learning in scenarios where centralized data collection is not feasible or desirable.

5.6 Answers to RQs

5.6.1 RQ1. Can participation in federated learning enhance model performance while ensuring data privacy?

Table 1 to Table 5 demonstrate that, across various tasks and models, multi-client federated learning consistently outperforms single-client training in code-related tasks. Take the defect detection task as an illustrative example, where we divided the dataset into two parts based on project names: one containing defect data from QEMU and the other from FFmpeg, a large-scale open-source project renowned for its rich and diverse defect information. Table 2 shows that despite the high quality of data in FFmpeg, federated learning still achieved superior results compared to single-client training solely on the FFmpeg defect dataset.

This observation underscores the advantages of federated learning in harnessing the collective power of multiple data sources while ensuring data privacy. This collaborative approach allows companies to leverage their respective datasets without disclosing sensitive information, thereby fostering a secure and privacy-preserving environment for the development of intelligent software engineering.

In addition to these performance advantages, federated learning also promotes a more sustainable and scalable model training ecosystem. By decentralizing the data collection and training processes, federated learning reduces the need for extensive computational resources typically required in centralized training, where all data must be aggregated in one place, such as the data center. This reduction in resource demands can be particularly beneficial in scenarios where training data is distributed across various geographical locations or organizational entities, such as multinational corporations or cross-border collaborations. Most importantly, it can obviously enhance the performance of the AI models in the aforementioned code-related tasks.

> **Answer for RQ1:** Compared to single client training, federated learning can enhance model performance while ensuring data privacy. The participant can get the benefit from federated learning.

5.6.2 RQ2. In code-related tasks, can federated learning achieve performance comparable to centralized training without disclosing data? Our experimental results indicate that, while there is indeed a performance gap between federated learning and centralized training in most tasks, especially evident in the code-to-text task, federated learning demonstrates considerable promise in specific scenarios. Notably, Table 5 shows that during the fine-tuning of large models, federated learning achieves result that closely resemble those obtained through centralized training. This suggests that federated learning has significant potential in this area. Furthermore, we observe that in specific code-related tasks, such as code search using CodeBERT, federated learning even surpasses the performance of centralized training. This finding implies that federated learning exhibits enhanced robustness in tasks like code search.

While federated learning may not always match the performance of centralized training across all code-related tasks, it offers a viable alternative that balances performance and privacy preservation, particularly in scenarios involving large model fine-tuning or specific tasks like code search.

Furthermore, the potential of federated learning to closely match centralized training outcomes highlights its relevance in environments where data privacy regulations, such as GDPR[1], limit the feasibility of centralized data aggregation. As privacy regulations become stricter worldwide, the ability of federated learning to ensure compliance while maintaining high performance could make it a preferred solution in the future of AI-based software engineering. Moreover, the reduced risk of data breaches due to the absence of central data storage further strengthens its appeal, especially in cases that data sensitivity is paramount. With the help of federated learning, the AI models can more effectively finish the tasks such as detecting code defects by adapting to different programming language and varying coding practices across teams. This flexibility ensures that the model is better suited to identify defects, regardless of the specific language or coding style used by different teams or individuals.

> **Answer for RQ2:** Federated learning has shown its potential to approach the performance of centralized training in some scenarios, but there still exists a gap.

5.6.3 RQ3. What impact does data heterogeneity have on the performance of federated learning in code-related tasks? Table 1 presents a detailed overview of the model performance in various data heterogeneity scenarios. When considering the *uniform distribution* strategy, we observe a significant drop in F1 score compared to the benchmark. On the other hand, the *label imbalance* strategy exhibits the lowest F1 score. This underscores the negative impact of label imbalance on model performance. Lastly, the *quantity imbalance* strategy maintains a slightly improved F1 score compared to uniform distribution in 2/4 aggregate strategies, indicating that the impact of quantity imbalance heterogeneity on clone detection model performance is relatively minor.

In summary, the result reveals that data heterogeneity can significantly influence the performance of models. Therefore, it is crucial to explore techniques and methods that can enhance the model's ability to handle diverse and imbalanced data distributions, thereby improving the overall accuracy and reliability of model.

[1] https://gdpr-info.eu/

Besides, as Table 2 and Table 5 shown, by employing an effective aggregation strategy that mitigates data heterogeneity, federated learning can attain performance closer to that of centralized training. There is potential for us to identify aggregation strategies that not only achieve performance close to centralized training but could possibly even surpass it.

Managing data heterogeneity is one of the most critical challenges in federated learning, as different clients often hold data with varying distributions, characteristics, and volumes. As the results suggest, label imbalances have a particularly negative impact on model performance, which implies the need for advanced balancing techniques or specialized algorithms to solve this problem. Future research could explore adaptive strategies for dynamically adjusting the importance of data from different clients, ensuring that even in highly imbalanced environments, federated learning models can maintain high accuracy. Additionally, due to the large size of models, smaller datasets may have a disinished impact. Therefore, ensuring that clients with smaller datasets are not overshadowed by those with larger data pools will be essential to promote fairness and consistency in model training.

> **Answer for RQ3:** Data heterogeneity, particularly imbalances in label distribution, can impact model performance. Exploring various aggregation strategies could be effective in mitigating this influence.

6 Challenges and Opportunities

6.1 Challenges

The vision of an open-source code model-sharing platform based on a decentralized federated learning system is to attract more entities to contribute their datasets and collaboratively develop and maintain code models. However, it still faces a series of challenges that need to be addressed. We list some important challenges and possible solutions.

Code Privacy Protection. A core advantage of federated learning is its protection of data privacy through local model training and sharing model weights. Nonetheless, the risk of data leakage still exists, such as threats from member inference attacks [43]. Unlike some traditional sensitive data, where anonymization can often be achieved by masking personal information, code data is uniquely sensitive for contributors—even in cases as minimal as a variable declaration. One current solution is fully homomorphic encryption [77], which allows model training on fully encrypted data and encrypts input data during the inference stage. Although this method is secure, it is computationally expensive. Another potential solution involves designing an intermediary medium to base the training of models and sharing of weights on.

Client variability and Model Stability. One significant challenge of federated learning in dynamic environments is managing client variability, such as the joining of new clients [30]. This variability can introduce risks of forgetting previously learned knowledge or destabilizing the global model. Techniques like incremental learning [67] and knowledge distillation [18] can mitigate this issue by enabling the global model to adapt to new clients while retaining important information from previous training. Addressing this challenge is crucial to ensuring the long-term stability and accuracy of federated models in decentralized systems. For instance, in the task of defect detection, we can simulate a real-world scenario where each participant possesses a distinctly different dataset by using heterogeneous datasets from multiple sources. To reflect the dynamic nature of federated learning in practice, we could introduce new clients with unique datasets into the collaborative training process. This setup would allow us to evaluate how the inclusion of new, diverse datasets impacts the overall model performance. To further investigate the impact of dynamic client participation on the stability and performance of federated models, we focus on the scenario where new clients with unique datasets are gradually introduced into the training process. These new datasets may represent distributions that are either disjoint or partially overlapping with the existing client data, simulating realistic scenarios where new participants join a federated network over time. Metrics such as changes in overall accuracy, variance in client-specific performance, and retention of previously learned knowledge will be evaluated to measure the stability and performance of federated models. Techniques like incremental learning and knowledge distillation will also be applied to ensure the model adapts effectively to new data while preserving information from earlier training. This experimental design provides a focused approach to studying dynamic client participation and its challenges in decentralized federated learning environments. Additionally, we could explore various strategies for updating the central model, focusing on how to best aggregate the knowledge learned from each client while preserving model stability and preventing performance degradation. By simulating these real-world conditions, we can better understand the challenges and opportunities associated with dynamic client participation and model updating in federated learning environments.

Reward Mechanisms. The development of open-source software often relies on the participation and maintenance of volunteers. However, not all individuals or entities are enthusiastic about participating in open-source projects. It is crucial to design reasonable incentive mechanisms to attract them more effectively to collaborate on open-source models. We can leverage the blockchain technique and create a "contribution token" to reward actively participating individuals and entities. For new users, we can provide initial system participation qualifications through an airdrop. The number of tokens held becomes a symbol of their contribution to the open-source community. This symbol may not directly translate into commercial benefits, but it at least ensures that their contribution is recorded on the blockchain and widely recognized and commemorated. Of course, more complex methods can be adopted, such as introducing DAO [14] governance and converting contribution tokens into voting power, making it an influential presence in the governance of the open-source community. Besides, in order to prevent some participants gaining disproportionate influence by accumulating a large number of tokens, a mechanism similar to block creation time in blockchain technology could be implemented. This would limit the rapid accumulation of tokens and ensure a more balanced distribution of influence across the open-source community, promoting fair and transparent governance.

Collaborative Interaction Protocols. When multiple parties collaborate to build and maintain open-source AI models, it is necessary to clarify the rules and protocols for collaborative work. These protocols should cover aspects such as model architecture, data formats, and methods for model updates. New contributors must follow the established rules. The open-source community needs a set of standards to guide the formulation of these protocols.

Copyright Issues. When multiple parties participate together, the ownership of the model may need to be clarified. If there are no clear ownership regulations, this could affect the enthusiasm of participants. To solve this problem, we believe there are two possible approaches: one is to establish copyright ownership rules while formulating collaborative interaction protocols; the other is to share copyrights based on contributions, where the contribution can be measured by the "contribution tokens" mentioned in the reward mechanism.

Security issue. If malicious participants contribute invalid model weights and data, it could seriously damage the open-source ecosystem and contaminate existing models [74]. The reward mechanism implemented based on smart contracts could also contain security vulnerabilities, possibly being exploited by hackers to maliciously acquire a large number of tokens [52]. Moreover, if there are particularly active entities in the open-source community, their continuous increase in token numbers might eventually lead them to a dominant position. It is crucial to design corresponding security mechanisms to prevent these security issues.

Energy Efficiency in Federated Learning. Although our proposed approach allows clients to train on their own data without needing to train on some public datasets, thus saving significant computational resources to some extent, federated learning systems still face the challenge of energy consumption, particularly when training large models across multiple clients. Energy-efficient federated learning designs could enhance the practicality of open-source AI-based software engineering tools. Techniques such as model compression [7], selective updates [11], and optimized communication protocols could significantly reduce both training time and energy consumption, making federated learning more feasible and cost-effective for diverse participants. Exploring these methods can lead to more sustainable models and lower barriers to participation.

The sustainability of AI-based SE tools. As AI models, particularly LLMs, become more widespread, their energy consumption and environmental impact cannot be ignored. Developing a governance framework that incorporates sustainability metrics is crucial. Organizations must measure energy efficiency, track software waste, and minimize the carbon footprint of their models [57]. Techniques such as model compression, selective updates, and more efficient data distribution methods can help reduce energy consumption in federated learning systems. Addressing these sustainability challenges will be essential for the long-term viability of AI-based software engineering.

6.2 Opportunities

Enhanced Collaboration through federated learning. Federated learning provides a unique opportunity for independent developers, companies and organizations to collaborate on AI-based software engineering models without sharing their private, sensitive data. This decentralized approach allows participants to collectively improve model performance while preserving privacy. By encouraging multi-party to collaboratively maintain the AI model, federated learning facilitates the integration of diverse, high-quality datasets across industries, which benefits to develop more robust and adaptable AI models. This form of collaboration is particularly valuable in areas such as defect detection, code generation and security enhancement, where the inclusion of diverse data can significantly improve outcomes.

Cost Reduction through Decentralized Collaboration. One of the key advantages of federated learning is the reduction of costs associated with AI model training. Instead of requiring vast computational resources to collect and process the huge centralized datasets, federated learning enable multiple participants collect and process their own data locally and then train the model on their own data. This decentralized approach reduces the need for redundant training on public datasets, which are usually used by multiple organizations as part of their AI model training. Instead, each client trains solely on their specific data, leading to more efficient use of computational resources. As a result, federated learning can lower both energy consumption and GPU costs, making AI development more sustainable and accessible for organizations with limited computational capacity.

Improved Privacy-Preserving Techniques. Federated learning enhances data privacy by aggregating only the updated parameters rather than the entire dataset. However, it still faces significant limitations when addressing malicious attacks or potential data leakage. Future work should prioritize developing more robust privacy-preserving techniques for both the local training process and the central aggregation process. For instance, methods like differential privacy [13] can provide stronger safeguards against privacy breaches.

Data handling for heterogeneity. Refining data handling for heterogeneity focuses on addressing the significant variations in data distribution, quality, and quantity across clients in federated learning systems. One potential approach is to use adaptive aggregation algorithms that dynamically adjust the weight of each client's update based on factors such as data quality, quantity, or token counts. This ensures that clients contributing more reliable or diverse data have a greater influence on the global model. Additionally, unified preprocessing techniques, such as normalization or feature alignment [61], can be applied before training to reduce inconsistencies in client data. These strategies collectively aim to improve the global model's ability to generalize across heterogeneous data while maintaining fairness and scalability in decentralized systems.

7 Conclusion

In this study, we delve into an emerging framework for the collaborative development of open-source code models, the code model-sharing mechanism based on federated learning. This framework aims to serve as a new paradigm for managing open-source AI-based SE models and to foster the joint construction and refinement of code models while ensuring data privacy protection. It addresses critical challenges such as data privacy, model evolution, and community engagement. Through this paradigm, we have detailed the specific impact of code data heterogeneity on model performance and identified key issues and challenges. The evaluation across different data distribution strategies demonstrates the potential of federated learning in improving the collaborative development of code models. Moreover, we discuss some potential challenges and solutions that might be faced. For example, we should intensify research on implementing more secure and reliable comprehensive encryption methods tailored to code characteristics. Regarding the contribution incentive mechanism for open-source models, we propose using a token-based contribution method to encourage more developers and participants to join this collaborative effort. At the same time, we also highlight the challenges of managing intellectual property rights for models in the implementation process. Besides, federated learning offers a promising pathway not only for collaborative AI model development but also for fostering sustainable software engineering practices. By decentralizing model training and reducing the energy burden on centralized systems, federated learning contributes to the long-term sustainability of AI-driven software tools, ensuring that organizations can balance innovation with environmental responsibility. This paper sets the foundation for future research, especially in handling real-world complexities, improving energy efficiency, and refining governance mechanisms to ensure fair and balanced participation in open-source projects. We look forward to further research and discussion in this field.

References

[1] 2022-11. *Chatgpt: Optimizing language models for dialogue.* https://chat.openai.com

[2] 2024. *Huggingface: The AI community building the future.* https://huggingface.co/

[3] Pekka Abrahamsson, Outi Salo, Jussi Ronkainen, and Juhani Warsta. 2017. Agile software development methods: Review and analysis. *arXiv preprint arXiv:1709.08439* (2017).

[4] Sadi Alawadi, Khalid Alkharabsheh, Fahed Alkhabbas, Victor Kebande, Feras M Awaysheh, and Fabio Palomba. 2023. Fedcsd: A federated learning based approach for code-smell detection. *arXiv preprint arXiv:2306.00038* (2023).

[5] Saleema Amershi, Andrew Begel, Christian Bird, Robert DeLine, Harald Gall, Ece Kamar, Nachiappan Nagappan, Besmira Nushi, and Thomas Zimmermann. 2019. Software engineering for machine learning: A case study. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP).* IEEE, 291–300.

[6] Andrea Arcuri and Xin Yao. 2008. A novel co-evolutionary approach to automatic software bug fixing. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence).* IEEE, 162–168.

[7] Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining.* 535–541.

[8] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374* (2021).

[9] Ting-Wu Chin, Ruizhou Ding, Cha Zhang, and Diana Marculescu. 2020. Towards efficient model compression via learned global ranking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition.* 1518–1528.

[10] Olivia Choudhury, Aris Gkoulalas-Divanis, Theodoros Salonidis, Issa Sylla, Yoonyoung Park, Grace Hsu, and Amar Das. 2019. Differential privacy-enabled federated learning for sensitive health data. *arXiv preprint arXiv:1910.02578* (2019).

[11] Kutluyil Dogancay and Oguz Tanrikulu. 2001. Adaptive filtering algorithms with selective partial updates. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing* 48, 8 (2001), 762–769.

[12] Rudresh Dwivedi, Devam Dave, Het Naik, Smiti Singhal, Rana Omer, Pankesh Patel, Bin Qian, Zhenyu Wen, Tejal Shah, Graham Morgan, et al. 2023. Explainable AI (XAI): Core ideas, techniques, and solutions. *Comput. Surveys* 55, 9 (2023), 1–33.

[13] Cynthia Dwork. 2006. Differential privacy. In *International colloquium on automata, languages, and programming.* Springer, 1–12.

[14] Youssef El Faqir, Javier Arroyo, and Samer Hassan. 2020. An overview of decentralized autonomous organizations on the blockchain. In *Proceedings of the 16th international symposium on open collaboration.* 1–8.

[15] Angela Fan, Beliz Gokkaya, Mark Harman, Mitya Lyubarskiy, Shubho Sengupta, Shin Yoo, and Jie M Zhang. 2023. Large language models for software engineering: Survey and open problems. *arXiv preprint arXiv:2310.03533* (2023).

[16] Amir Feder, Nadav Oved, Uri Shalit, and Roi Reichart. 2021. CausaLM: Causal model explanation through counterfactual language models. *Computational Linguistics* 47, 2 (2021), 333–386.

[17] Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, et al. 2020. Codebert: A pre-trained model for programming and natural languages. *arXiv preprint arXiv:2002.08155* (2020).

[18] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. 2021. Knowledge distillation: A survey. *International Journal of Computer Vision* 129, 6 (2021), 1789–1819.

[19] Daya Guo, Shuo Ren, Shuai Lu, Zhangyin Feng, Duyu Tang, Shujie Liu, Long Zhou, Nan Duan, Alexey Svyatkovskiy, Shengyu Fu, et al. 2020. Graphcodebert: Pre-training code representations with data flow. *arXiv preprint arXiv:2009.08366* (2020).

[20] Sirui Hong, Xiawu Zheng, Jonathan Chen, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, et al. 2023. Metagpt: Meta programming for multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352* (2023).

[21] Xinyi Hou, Yanjie Zhao, Yue Liu, Zhou Yang, Kailong Wang, Li Li, Xiapu Luo, David Lo, John Grundy, and Haoyu Wang. 2024. Large Language Models for Software Engineering: A Systematic Literature Review. *ACM Trans. Softw. Eng. Methodol.* (Sept. 2024). https://doi.org/10.1145/3695988 Just Accepted.

[22] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *International conference on machine learning.* PMLR, 2790–2799.

[23] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685* (2021).

[24] Hamel Husain, Ho-Hsiang Wu, Tiferet Gazit, Miltiadis Allamanis, and Marc Brockschmidt. 2019. Codesearchnet challenge: Evaluating the state of semantic code search. *arXiv preprint arXiv:1909.09436* (2019).

[25] Johannes Rude Jensen, Victor von Wachter, and Omri Ross. 2021. How decentralized is the governance of blockchain-based finance: Empirical evidence from four governance token distributions. *arXiv preprint arXiv:2102.10096* (2021).

[26] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. 2022. Visual prompt tuning. In *European Conference on Computer Vision*. Springer, 709–727.

[27] Nan Jiang, Thibaud Lutellier, and Lin Tan. 2021. Cure: Code-aware neural machine translation for automatic program repair. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 1161–1173.

[28] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361* (2020).

[29] Donghyun Kim, Kaihong Wang, Stan Sclaroff, and Kate Saenko. 2022. A broad study of pre-training for domain generalization and adaptation. In *European Conference on Computer Vision*. Springer, 621–638.

[30] Yeongwoo Kim, Ezeddin Al Hakim, Johan Haraldson, Henrik Eriksson, José Mairton B da Silva, and Carlo Fischione. 2021. Dynamic clustering in federated learning. In *ICC 2021-IEEE International Conference on Communications*. IEEE, 1–6.

[31] Jakub Konecnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. 2016. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492* 8 (2016).

[32] Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, et al. 2023. Starcoder: may the source be with you! *arXiv preprint arXiv:2305.06161* (2023).

[33] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems* 2 (2020), 429–450.

[34] Yuzheng Li, Chuan Chen, Nan Liu, Huawei Huang, Zibin Zheng, and Qiang Yan. 2020. A blockchain-based decentralized federated learning framework with committee consensus. *IEEE Network* 35, 1 (2020), 234–241.

[35] Zhen Li, Deqing Zou, Shouhuai Xu, Xinyu Ou, Hai Jin, Sujuan Wang, Zhijun Deng, and Yuyi Zhong. 2018. Vuldeepecker: A deep learning-based system for vulnerability detection. *arXiv preprint arXiv:1801.01681* (2018).

[36] Shangqing Liu, Yanzhou Li, Xiaofei Xie, and Yang Liu. 2022. Commitbart: A large pre-trained model for github commits. *arXiv preprint arXiv:2208.08100* (2022).

[37] Shuai Lu, Daya Guo, Shuo Ren, Junjie Huang, Alexey Svyatkovskiy, Ambrosio Blanco, Colin Clement, Dawn Drain, Daxin Jiang, Duyu Tang, et al. 2021. Codexglue: A machine learning benchmark dataset for code understanding and generation. *arXiv preprint arXiv:2102.04664* (2021).

[38] Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. 2023. Wizardcoder: Empowering code large language models with evol-instruct. *arXiv preprint arXiv:2306.08568* (2023).

[39] Wei Ma, Shangqing Liu, Zhihao Lin, Wenhan Wang, Qiang Hu, Ye Liu, Cen Zhang, Liming Nie, Li Li, and Yang Liu. 2023. LMs: Understanding Code Syntax and Semantics for Code Analysis. *arXiv preprint arXiv:2305.12138* (2023).

[40] Wei Ma, Shangqing Liu, Mengjie Zhao, Xiaofei Xie, Wenhang Wang, Qiang Hu, Jie Zhang, and Yang Liu. 2024. Unveiling Code Pre-Trained Models: Investigating Syntax and Semantics Capacities. *ACM Trans. Softw. Eng. Methodol.* 33, 7, Article 169 (Aug. 2024), 29 pages. https://doi.org/10.1145/3664606

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

[42] H Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. 2016. Federated learning of deep networks using model averaging. *arXiv preprint arXiv:1602.05629* 2 (2016), 2.

[43] Milad Nasr, Reza Shokri, and Amir Houmansadr. 2019. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE symposium on security and privacy (SP)*. IEEE, 739–753.

[44] Henry YT Ngan, Grantham KH Pang, and Nelson HC Yung. 2011. Automated fabric defect detection—A review. *Image and vision computing* 29, 7 (2011), 442–458.

[45] Eirini Ntoutsi, Pavlos Fafalios, Ujwal Gadiraju, Vasileios Iosifidis, Wolfgang Nejdl, Maria-Esther Vidal, Salvatore Ruggieri, Franco Turini, Symeon Papadopoulos, Emmanouil Krasanakis, et al. 2020. Bias in data-driven artificial intelligence systems—An introductory survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 10, 3 (2020), e1356.

[46] Open-Source AI Models. 2024. *Open-Source AI Models*. https://github.com/mathieu0905/collaborative_software_learning

[47] Sinno Jialin Pan and Qiang Yang. 2009. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* 22, 10 (2009), 1345–1359.

[48] Lakshmish Ramaswamy, Bugra Gedik, and Ling Liu. 2005. A distributed approach to node clustering in decentralized peer-to-peer networks. *IEEE Transactions on Parallel and Distributed Systems* 16, 9 (2005), 814–829.

[49] David Rolnick, Priya L Donti, Lynn H Kaack, Kelly Kochanski, Alexandre Lacoste, Kris Sankaran, Andrew Slavin Ross, Nikola Milojevic-Dupont, Natasha Jaques, Anna Waldman-Brown, et al. 2022. Tackling climate change with machine learning. *ACM Computing Surveys (CSUR)* 55, 2 (2022), 1–96.

[50] Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950* (2023).

[51] Snehal Satish, Geeta Sandeep Nadella, Karthik Meduri, and Hari Gonaygunta. 2022. Collaborative Machine Learning without Centralized Training Data for Federated Learning. *International Machine Learning Journal and Computer Engineering* 5, 5 (2022), 1–14.

[52] Sarwar Sayeed, Hector Marco-Gisbert, and Tom Caira. 2020. Smart Contract: Attacks and Protections. *IEEE Access* 8 (2020), 24416–24427. https://doi.org/10.1109/ACCESS.2020.2970495

[53] Koustuv Sinha, Robin Jia, Dieuwke Hupkes, Joelle Pineau, Adina Williams, and Douwe Kiela. 2021. Masked language modeling and the distributional hypothesis: Order word matters pre-training for little. *arXiv preprint arXiv:2104.06644* (2021).

[54] Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2020. Energy and policy considerations for modern deep learning research. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 13693–13696.

[55] Evi Suryawati and Kamisah Osman. 2017. Contextual learning: Innovative approach towards the development of students' scientific attitude and natural science performance. *Eurasia Journal of mathematics, science and technology education* 14, 1 (2017), 61–76.

[56] Edna Chebet Too, Li Yujian, Sam Njuki, and Liu Yingchun. 2019. A comparative study of fine-tuning deep learning models for plant disease identification. *Computers and Electronics in Agriculture* 161 (2019), 272–279.

[57] Mueen Uddin and Azizah Abdul Rahman. 2012. Energy efficiency and low carbon enabler green IT framework for data centers considering green metrics. *Renewable and Sustainable Energy Reviews* 16, 6 (2012), 4078–4094.

[58] Aimee Van Wynsberghe. 2021. Sustainable AI: AI for sustainability and the sustainability of AI. *AI and Ethics* 1, 3 (2021), 213–218.

[59] Julian Von der Mosel, Alexander Trautsch, and Steffen Herbold. 2022. On the validity of pre-trained transformers for natural language processing in the software engineering domain. *IEEE Transactions on Software Engineering* 49, 4 (2022), 1487–1507.

[60] Junjie Wang, Yuchao Huang, Chunyang Chen, Zhe Liu, Song Wang, and Qing Wang. 2024. Software testing with large language models: Survey, landscape, and vision. *IEEE Transactions on Software Engineering* (2024).

[61] Lidong Wang. 2017. Heterogeneous data and big data analytics. *Automatic Control and Information Sciences* 3, 1 (2017), 8–15.

[62] Yue Wang, Hung Le, Akhilesh Deepak Gotmare, Nghi DQ Bui, Junnan Li, and Steven CH Hoi. 2023. Codet5+: Open code large language models for code understanding and generation. *arXiv preprint arXiv:2305.07922* (2023).

[63] Yue Wang, Weishi Wang, Shafiq Joty, and Steven C.H. Hoi. 2021. CodeT5: Identifier-aware Unified Pre-trained Encoder-Decoder Models for Code Understanding and Generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (Eds.). Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 8696–8708. https://doi.org/10.18653/v1/2021.emnlp-main.685

[64] Zhilin Wang and Qin Hu. 2021. Blockchain-based federated learning: A comprehensive survey. *arXiv preprint arXiv:2110.02182* (2021).

[65] Erik Wibbels. 2005. Decentralized governance, constitution formation, and redistribution. *Constitutional Political Economy* 16 (2005), 161–188.

[66] Carole-Jean Wu, Ramya Raghavendra, Udit Gupta, Bilge Acun, Newsha Ardalani, Kiwan Maeng, Gloria Chang, Fiona Aga, Jinshi Huang, Charles Bai, et al. 2022. Sustainable ai: Environmental implications, challenges and opportunities. *Proceedings of Machine Learning and Systems* 4 (2022), 795–813.

[67] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. 2019. Large scale incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 374–382.

[68] Miao Xiong, Zhiyuan Hu, Xinyang Lu, Yifei Li, Jie Fu, Junxian He, and Bryan Hooi. 2023. Can llms express their uncertainty? an empirical evaluation of confidence elicitation in llms. *arXiv preprint arXiv:2306.13063* (2023).

[69] Frank F Xu, Uri Alon, Graham Neubig, and Vincent Josua Hellendoorn. 2022. A systematic evaluation of large language models of code. In *Proceedings of the 6th ACM SIGPLAN International Symposium on Machine Programming*. 1–10.

[70] Hanxiang Xu, Wei Ma, Ting Zhou, Yanjie Zhao, Kai Chen, Qiang Hu, Yang Liu, and Haoyu Wang. 2024. A Code Knowledge Graph-Enhanced System for LLM-Based Fuzz Driver Generation. *arXiv preprint arXiv:2411.11532* (2024).

[71] John Yang, Carlos E. Jimenez, Alexander Wettig, Shunyu Yao, Karthik Narasimhan, and Ofir Press. 2024. SWE-agent: Agent Computer Interfaces Enable Software Engineering Language Models.

[72] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)* 10, 2 (2019), 1–19.

[73] Yanming Yang, Xing Hu, Zhipeng Gao, Jinfu Chen, Chao Ni, Xin Xia, and David Lo. 2024. Federated Learning for Software Engineering: A Case Study of Code Clone Detection and Defect Prediction. *IEEE Trans. Softw. Eng.* 50, 2 (jan 2024), 296–321. https://doi.org/10.1109/TSE.2023.3347898

[74] Yifan Yao, Jinhao Duan, Kaidi Xu, Yuanfang Cai, Zhibo Sun, and Yue Zhang. 2024. A survey on large language model (llm) security and privacy: The good, the bad, and the ugly. *High-Confidence Computing* (2024), 100211.

[75] Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. 2018. Byzantine-robust distributed learning: Towards optimal statistical rates. In *International Conference on Machine Learning*. Pmlr, 5650–5659.

[76] Zhiqiang Yuan, Mingwei Liu, Shiji Ding, Kaixin Wang, Yixuan Chen, Xin Peng, and Yiling Lou. 2024. Evaluating and Improving ChatGPT for Unit Test Generation. *Proc. ACM Softw. Eng.* 1, FSE, Article 76 (July 2024), 24 pages. https://doi.org/10.1145/3660783

[77] Chen Zhang, Yu Xie, Hang Bai, Bin Yu, Weihong Li, and Yuan Gao. 2021. A survey on federated learning. , 106775 pages.

[78] Cen Zhang, Yaowen Zheng, Mingqiang Bai, Yeting Li, Wei Ma, Xiaofei Xie, Yuekang Li, Limin Sun, and Yang Liu. 2024. How Effective Are They? Exploring Large Language Model Based Fuzz Driver Generation. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis* (Vienna, Austria) *(ISSTA 2024)*. Association for Computing Machinery, New York, NY, USA, 1223–1235. https://doi.org/10.1145/3650212.3680355

[79] Yaqin Zhou, Shangqing Liu, Jingkai Siow, Xiaoning Du, and Yang Liu. 2019. Devign: Effective vulnerability identification by learning comprehensive program semantics via graph neural networks. *Advances in neural information processing systems* 32 (2019).

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