Semina Ciências Exatas e Tecnológicas

e-ISSN 1679-0375
COMPUTER SCIENCE

ORIGINAL ARTICLE DOI 10.5433/1679-0375.2026.v47.53937

iThenticate

A Comparative Study of Federated Learning Frameworks for IoT-Driven Smart Cities

Um Estudo Comparativo de Frameworks de Aprendizado Federado para Cidades Inteligentes baseadas em Internet das Coisas

Leandro di Lauro[1], Bruno Bogaz Zarpelão[2], Sylvio Barbon Junior[3]

Received: October 3, 2025 *Received in revised form: February 21, 2026* *Accepted: March 11, 2026* *Available online: April 22, 2026*

ABSTRACT

The integration of Machine Learning (ML) with smart cities offers promising solutions for analyzing IoT (Internet of Things) data. However, conventional ML approaches often struggle to balance accuracy and computational costs, particularly in heterogeneous and resource-constrained IoT environments. Federated Learning (FL) has emerged as an alternative, enabling privacy-preserving collaborative learning across distributed devices while reducing centralized data collection. Over the past decade, both industry and academia have contributed to a growing ecosystem of FL frameworks with distinct functionalities and design choices. Despite this progress, research on FL in smart cities remains limited, making it difficult to assess framework suitability for real-world use cases. This paper presents a comparative analysis of FL frameworks in engineering-oriented smart city scenarios. We evaluate representative frameworks, focusing on Flower and NVFlare, using datasets derived from IoT infrastructures. The results highlight the trade-offs of FL compared to conventional ML, offering insights for researchers, engineers, and decision-makers.

keywords smart city, internet of things, machine learning, federated learning, privacy

RESUMO

A integração de Aprendizado de Máquina (AM) com cidades inteligentes oferece soluções promissoras para a análise de dados de *Internet of Things* (IoT). No entanto, abordagens convencionais de AM frequentemente enfrentam dificuldades para equilibrar acurácia e custo computacional, especialmente em ambientes de IoT heterogêneos e com restrições de recursos. O Aprendizado Federado (AF) surge como uma alternativa, permitindo o aprendizado colaborativo com preservação de privacidade, ao mesmo tempo em que reduz a necessidade de coleta centralizada de dados. Na última década, tanto a indústria quanto a academia contribuíram para um ecossistema crescente de *frameworks* de AF, com diferentes funcionalidades e escolhas de *design*. Apesar desse avanço, a pesquisa sobre AF em cidades inteligentes ainda é limitada, dificultando a avaliação da adequação dos *frameworks* para aplicações reais. Este artigo apresenta uma análise comparativa de *frameworks* de AF em cenários de cidades inteligentes. Avaliamos *frameworks* representativos, com foco em Flower e NVFlare, utilizando conjuntos de dados derivados de infraestruturas de IoT. Os resultados evidenciam os equilíbrios entre ganhos e limitações do AF em comparação às abordagens convencionais de AM, oferecendo subsídios para pesquisadores, engenheiros e gestores.

palavras-chave cidades inteligentes, internet das coisas, aprendizado de máquina, aprendizado federado, privacidade

[1]MSc in Electronic Engineering, Università degli Studi di Trieste, Trieste, Italy. leandro.dilauro.units@gmail.com
[2]Prof. Dr., Computer Science Department, State University of Londrina, Londrina, PR, Brazil. brunozarpelao@uel.br
[3]Prof. Dr., Department of Engineering and Architecture, Università degli Studi di Trieste, Trieste, Italy. sylvio.barbonjunior@units.it

Introduction

Among the various real-life applications of Artificial Intelligence (AI) in contemporary society, the use of Machine Learning (ML) in smart city solutions involves analyzing Internet of Things (IoT) data, addressing security challenges and integrating diverse technologies. However, several challenges remain, including ensuring system reliability, achieving scalability and enabling effective integration across different infrastructure layers (Ullah et al., 2024).

Current ML tools for smart cities often fail to balance the trade-off between accuracy and computational cost in real-life applications. Furthermore, the integration of recent IoT technologies enhances smart city capabilities but can further aggravate the computational burden and resource constraints (Majeed et al., 2021). Several approaches have been proposed to address this gap, particularly through novel strategies and architectures, among which Federated Learning (FL) stands out as one of the most prominent.

The concept of FL can be traced back to at least 2015 (Konečný et al., 2015) as a practical approach for achieving privacy-preserving collaborative learning among multiple parties. Instead of collecting data from each party, models are trained locally on each device, and only the model updates or aggregated gradients are exchanged. This approach helps preserve data privacy thanks to its decentralized structure, while leveraging the collective knowledge across distributed devices.

Since then, FL frameworks have rapidly advanced, keeping pace with developments in distributed computing, privacy-enhancing technologies, and ML paradigms. A lot of major tech companies, such as Google and NVIDIA, started to develop their own FL framework given the potential of this recent technique. This led to the creation of FL frameworks named respectively: FATE (Liu et al., 2021), TensorFlow Federated (TFF), NVIDIAFlare (NVFlare) (Roth et al., 2022), PySyft (Ziller et al., 2021), OpenFL (Foley et al., 2022), FederatedScope(FS) (Xie et al., 2022) and substra (Galtier & Marini, 2019).

In addition, FL frameworks have seen significant contributions from academic institutions and research communities, driving to the creation of other frameworks called: Flower (Beutel et al., 2022), FedML (He et al., 2020), vantage6 (Moncada-Torres et al., 2021), APPFL (Ryu et al., 2022), Flame (Cho et al., 2022) and FedN (Ekmefjord et al., 2022).

The growing prominence of FL frameworks has led to a diverse array of options, each characterized by distinctive features, design principles, and compatibility with various ML platforms.

As FL is a relatively recent technique, the number of studies addressing its application in IoT and smart city contexts remains limited. Consequently, it is challenging to determine the most suitable FL framework for a given use case and to evaluate its performance when applied to smart city scenario. This lack of evidence hinders practioners in selecting appropriate frameworks and in estimating achievable performance under different infrastructures compared to conventional ML. In particular, limited research exists on the performance of FL in Deep Learning (DL) (Heidari et al., 2022) applied to tabular data, including both general records of daily human activities and domain-specific measurements from automotive or power system sensors.

The aim of this paper is to present an overview of the functionalities of currently available open-source FL frameworks, with an emphasis on their strengths and limitations across different engineering use cases. To achieve this goal, we conduct a two-phase comparative evaluation.

In Phase 1, multiple FL frameworks are analyzed according to criteria such as data heterogeneity and partitioning, communication topologies, compatibility with development environments, deployment readiness, and privacy and security features. This analysis motivates the selection of Flower and NVFlare as representative implementations. These frameworks are chosen due to their maturity, quality of documentation, ease of integration, support for IID and non-IID horizontally partitioned data, readiness for commercial use, and availability of built-in privacy and security mechanisms.

In Phase 2, the selected frameworks are experimentally compared with a centralized baseline across different case studies. The evaluation relies on quantitative metrics for model performance, namely accuracy, F1-score, and loss, and for hardware resource usage, namely execution time, CPU usage, and RAM usage. Scalability with respect to the number of participating clients and the efficiency impact of privacy-preserving techniques are also examined.

The main contributions of this work are:

- An overview and comparative analysis of functionalities, strengths, and limitations of selected FL frameworks in engineering-oriented smart city use cases.
- A performance evaluation of FL against conventional DL approaches using datasets from IoT and smart city environments.
- An experimental study employing two representative FL frameworks, Flower and NVFlare, selected and applied according to the requirements of specific case studies.
- Insights into the advantages and drawbacks of FL in distributed learning scenarios, providing guidance for its practical adoption in urban infrastructure applications.

Federated learning in smart cities

Over the past twenty years, the concept of Smart Cities has seen a significant rise in interest (Gracias et al., 2023). This surge is largely due to the increased urban population, which has put pressure on the environment, infrastructure, and resources. To address these challenges, there has been a significant push towards transforming existing cities into smart cities, with a lot of the bigger cities leading the way by developing smart cities projects. Smart cities differ from traditional cities in several ways, such as in their diversity of users, high levels of user engagement, dependence on resource-constrained IoT devices, high mobility of goods and technology, enhanced connectivity, and scalable development. Implementing information and communication technologies (ICT) and IoT in smart cities enhances the monitoring and integration of various infrastructure systems, leading to more efficient operational decisions (Nguyen et al., 2024). Consequently, smart cities may become more efficient, liveable, and sustainable, offering advancements in smart housing, transportation, healthcare, and governance. These environments require robust cybersecurity measures and decentralized approaches to data processing and learning, due to the scale, heterogeneity, and distributed nature of their connected systems.

The IoT-based architecture of smart cities requires effective solutions for security and privacy concerns (Cui et al., 2018). This architecture comprises four layers: perception, network, support, and applications. Data moving between these layers faces numerous security and privacy risks, including unauthorized access, tampering, and data leaks. These security risks can undermine the reliability and effectiveness of intelligent services. For instance, in March 2018, the City of Atlanta was hit by a severe ransomware attack that encrypted data and disrupted municipal services, leading to over $17 million in recovery costs (Blinder & Perlroth, 2018). Despite implementing various security measures, many smart city systems remain vulnerable due to the low computational power of devices in the perception layer and the complexity and dynamism of IoT systems.

Smart systems in smart cities generate big data that is essential for training AI systems. However, securing this data has become a major challenge, affecting individuals, businesses, and researchers. Governments have introduced regulations to protect data privacy, complicating the development of AI models in centralized systems. Privacy and security standards such as Health Insurance Portability and Accountability Act of 1996 (Annas, 2003) (HIPAA), Data protection Impact Assessment (Georgiou & Lambrinoudakis, 2021) (DPIA) and the European Union's General Data Protection Regulation (The European Parliament and the Council of the European Union, 2016) (GDPR) are some of the most important regulations in this domain. While these regulations play a vital role in safeguarding privacy and security, it is important to acknowledge that they might not provide a comprehensive solution to mitigate all potential attacks. To address these issues, a new decentralized AI approach known as FL has been developed.

FL has surfaced as a groundbreaking shift in how data privacy and confidentiality issues are addressed within ML. This approach marks a departure from the conventional centralized models by facilitating a collaborative effort among various participants to develop a unified model without the need to centralize sensitive data. By empowering individual devices or participants to train models locally and only share model improvements with a central hub, FL presents a privacy-aware alternative while harnessing the collective wisdom of scattered data points.

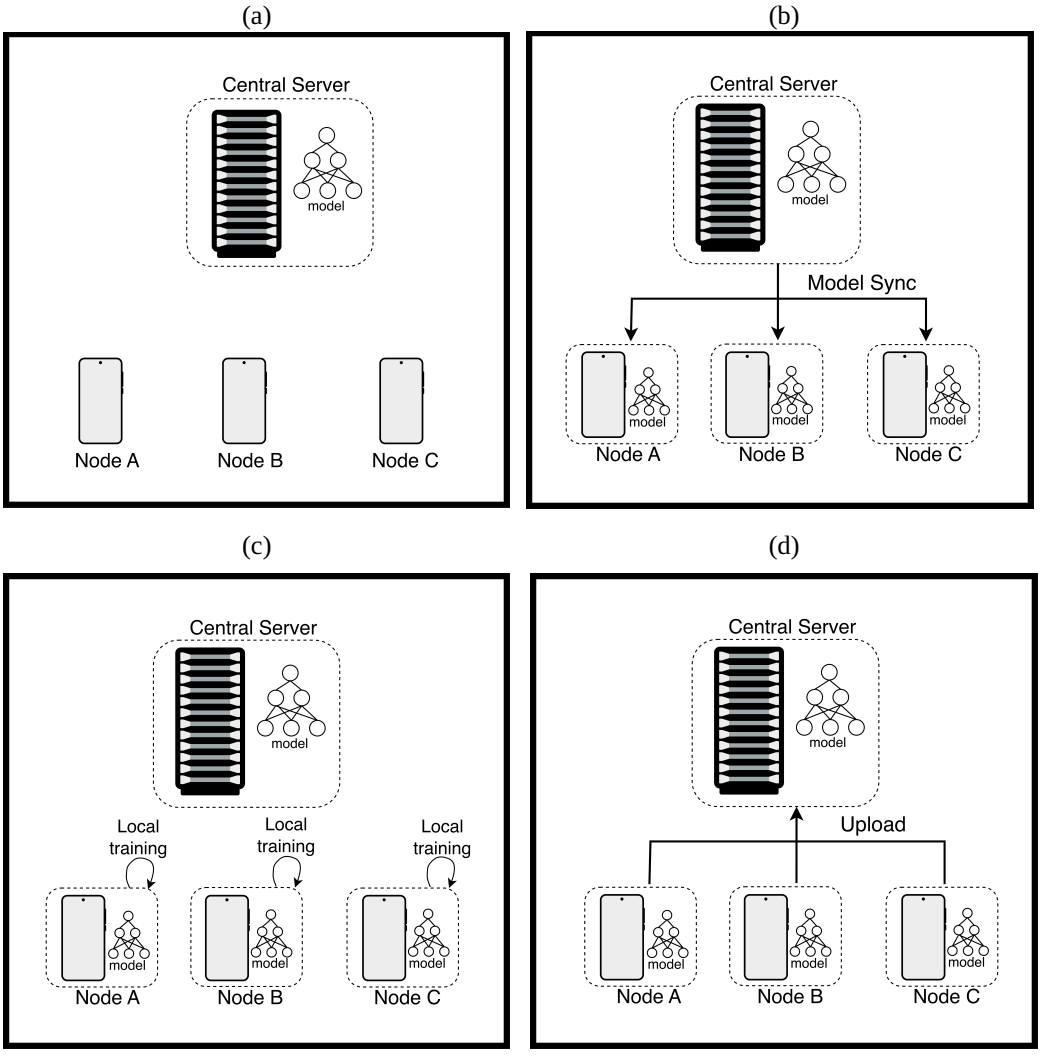
The FL process (Banabilah et al., 2022) involves a central authority hosting a global ML model, which is then distributed to participants or users on their local devices. This global model acts as a starting point for each participant, who uses their own data to train the model locally. This setup ensures that personal data never leaves the user's device, which helps maintain privacy and security.

After training locally, participants only send updated model parameters back to the central server, which aggregates these updates to refine the global model.

This cycle, with the updated global model being redistributed for further training, repeats iteratively, enhancing the model's accuracy over time while keeping individual data confidential.

In Figure 1, the illustration of a typical FL process highlights a central orchestrator setup, emphasizing the collaborative nature of this learning approach. The rise of FL has led to the creation of various frameworks designed to simplify the adoption and execution of this decentralized ML paradigm across different sectors. These frameworks are built to handle diverse data types, model structures, and communication protocols, simplifying FL's complexities and making it more accessible to a wider audience without deep technical knowledge.

Figure 1 - A typical FL architecture: (a) The central server selects a model to be trained; (b) The central server sends the initial model to multiple nodes; (c) The client nodes train the model locally using their own datasets; (d) The client nodes upload their local models to the central server, which combines the updates into one global model without accessing the data.



Moreover, these FL frameworks offer significant security and privacy advantages, tackling potential risks like data breaches, model inversion attacks, and other vulnerabilities (Hasan, 2023). They incorporate sophisticated security measures such as encryption, authentication, and access control, laying the groundwork for a secure collaborative ML environment.

Related work

In recent years, the integration of FL into smart city applications has emerged as a promising avenue for enhancing urban systems' efficiency while preserving users data privacy and security. An analysis on the application of FL within smart cities is provided by Jiang et al. (2020), where they present an overview of smart city sensing and its current challenges followed by the potential of FL in addressing those challenges. A comprehensive discussion of the state-of-the-art methods for FL is provided along with an in-depth discussion on the applicability of this technique in the field of Mobile Crowdsensing and showing advantage and disadvantages in using FL in this scenario.

The study by Alla and Thangarasu (2023) tackled the challenges of privacy and security in leveraging IoT-generated big data within smart city applications. They introduced a strategy that merges FL with the particle swarm optimization (PSO) algorithm, aiming to boost prediction precision while safeguarding data confidentiality. By decentralizing the learning process across IoT devices, the FL method ensures data privacy and harnesses the power of collective intelligence. Thanks to this approach, they not only was able to achieve outstanding classification accuracy but also upheld data privacy and reduced privacy and security vulnerabilities. This research enriched the understanding of privacy and security issues in big data analytics, offering valuable perspectives for smart city initiatives and the handling of IoT-generated data.

With their work, Djenouri et al. (2023) explored a cutting-edge federated DL method for forecasting urban traffic flow that combines graph learning with a trusted authority mechanism. They conducted a comprehensive evaluation of their FL-based framework, including a case study on forecasting future traffic flows using multiple datasets. They also compared it against various baseline techniques. The results demonstrated that their proposed framework significantly outperformed the baseline methods, especially in scenarios where the graph contained a large number of nodes, obtaining also an average precision of the developed model of 84%, while the baseline solutions did not exceed 77%.

The development of FL solutions and infrastructures demands specialized expertise and extensive programming, as well as high-performance devices capable of training data locally. To address this gap, Valente et al. (2023) introduced an architecture for a lightweight container-based solution, designed to offer a variety of ML algorithms for creating prediction engines on edge devices. This architecture also encompassed the primary options for algorithms/models for the aggregation and refinement of models on a central server. Their proposed framework facilitated the swift development of containerized testbeds for the initial evaluation phase of ML and aggregation algorithms, as well as their subsequent deployment in real production environments. They showcased the effectiveness of their approach by estimating vehicle mobility into and out of the city of Aveiro, utilizing real data gathered by the communications and sensing infrastructure.

Since the most important issue when we talk about FL is the privacy and security of individual users' data, several studies and scientific articles have been written about it. For example, the study by Al-Huthaifi et al. (2023) offers a detailed examination of the application of FL to enhance privacy and security within smart city systems, including transportation, healthcare, and communication sectors. They thoroughly discussed the advantages, disadvantages, ongoing research challenges, and future prospects of implementing FL in smart cities environments. The research found that the current FL systems needed extensive testing against various types of attacks to further bolster data protection and performance within smart city frameworks.

To enhance data privacy and security in smart cities, Kuang and Chen (2023) proposed the use of function encryption combined with differential privacy techniques within a FL framework. Specifically, they introduced the FE-BDP model, which incorporates Bayesian Differential Privacy to protect aggregation weights, and the FE-LDP model, which integrates Local Differential Privacy for decentralized key security management. Their experimental results demonstrate that the FE-BDP model maintains high accuracy, while the FE-LDP model achieves significantly lower loss compared to other approaches, indicating the effectiveness of their methods in preserving privacy without compromising model performance.

Another research about data privacy and security was made by Wang et al. (2023). They introduced a privacy-centric FL approach utilizing homomorphic encryption (PPFLHE). Both theoretical evaluations and practical experiments demonstrated that this method not only ensures robust data utility and a classification accuracy of 81.53% but also minimised communication delays. This was achieved while maintaining stringent privacy safeguards, setting it apart from contemporary techniques in the field.

This paper presents a novel contribution to the smart cities domain by exploring two prominent FL frameworks and emphasizing the advantages of FL over traditional ML in enhancing privacy, security, and efficiency. Central to the study is the concept of node influence, which examines the effects of local versus centralized training in distributed environments. The research applies DL models across all FL experiments and incorporates IoT devices as key data sources, enabling high-dimensional urban analytics. Additionally, the study evaluates the use of homomorphic encryption to secure data during aggregation while preserving its analytical utility, offering a balanced perspective on its practicality. Altogether, the work demonstrates how the combination of FL, DL, and IoT can drive the development of more robust, efficient, and privacy-preserving smart city infrastructures.

Federated learning frameworks

This section examines the features of different FL frameworks and highlights the need for careful selection when deploying FL solutions in complex urban environments. For this study, we focused on frameworks that have source code publicly available on GitHub (search term: “federated learning”) (Karimireddy et al., 2023). To reduce the large and rapidly expanding set of options, we applied the following preliminary screening criteria:

- Active development: We give preference to frameworks that consistently show commits, reflecting continuous development and maintenance, and that have also released at least one release in the last year. This criterion disqualifies PaddleFL, FedScale, EasyFL, Federated Learning Simulator (FLSim), International Business Machines FL (IBMFL) and many other purely academic framework implementations.
- Open-source code: We exclusively evaluate frameworks with publicly available source code hosted on GitHub. This excludes services such as the Google Cloud FL Service, which hasn’t disclosed its source code.
- Broad applications: We exclude frameworks that are highly specialized for specific applications, such as FedTree by the National University of Singapore, designed specifically for tree-based models and other minor repositories developed for specific applications.
- FL-centric libraries: Our focus is exclusively on libraries primarily designed for FL, setting aside those with a blockchain emphasis, like Oasis protocols from Oasis, and others aimed at scalable distributed training, such as Apache Spark by Databricks, which might be adapted for FL use.

After the application of the above-mentioned criteria, a large number of frameworks remained, including several that have been developed only recently. We outline the remaining frameworks below, along with the primary organizations to which the majority of their contributors are affiliated. It’s important to note that, given the open-source nature of these projects, many individuals from various organizations may have also made contributions to each project. The selected framework are as following:

1. **FATE**: by Webank
2. **TFF**: by Google
3. **Flower**: by Univeristy of Cambridge
4. **Vantage6**: by Netherlands Comprehensive Cancer Organisation (IKNL)
5. **FS**: by Alibaba
6. **NVFlare**: by NVIDIA
7. **APPFL**: by Argonne National Laboratory
8. **FedML**: by FedML.ai
9. **OpenFL**: by Intel
10. **Flame**: by Cisco
11. **FedN**: by Scaleout Systems.
12. **Substra**: by Owkin
13. **PySyft**: by OpenMinded

Through the various selection criteria applied, we were able to obtain this list of 13 frameworks in order to provide a comprehensive overview of FL frameworks. Although this is an overview and not an in-depth comparison of the various frameworks, it is nevertheless a very complicated task due to the continuous evolution of frameworks that may integrate new features from month to month, thus making it difficult to have an up-to-date version of the literature on the subject.

To support a comprehensive evaluation of FL frameworks for smart city applications, we identified five additional key criteria: handling of data heterogeneity and partitioning, communication topologies, compatibility and integration with existing tools, readiness for commercial use, and privacy and security features. These aspects provide a structured basis for comparing frameworks according to their technical capabilities and practical applicability. We use the term high versatility to denote: (i) support for heterogeneous FL settings and system/data heterogeneity; (ii) interoperability with different ML training stacks and workflows; and (iii) availability of deployment- and governance-oriented building blocks (e.g., privacy/security options) required in real-world smart-city/IoT scenarios. Under this operational definition, Flower is versatile because it is explicitly designed to study large-scale FL workloads under richly heterogeneous device scenarios and to enable seamless migration from simulation to real devices (Beutel et al., 2022). Similarly, NVFlare targets simulation-to-real-world FL deployments and supports multiple training libraries (e.g., PyTorch, TensorFlow, XGBoost, NumPy), while also providing privacy-preserving options such as homomorphic encryption and differential privacy (Roth et al., 2022). Our interpretation of versatility as a multi-dimensional selection criterion is consistent with recent literature-based comparative analyses of open-source FL frameworks (Riedel et al., 2024). Each of these criteria is discussed in detail in the following subsections.

Data heterogeneity and partitioning handling

In this section, we discuss the challenges posed by heterogeneous data distributions among devices, covering both IID (independently and identically distributed) (Arafeh et al., 2022) and non-IID (Zhu et al., 2021) data scenarios. Furthermore, we delve into the various types of data partitioning techniques, including vertical, horizontal, and hybrid partitioning, and explore their significance in FL settings. Data heterogeneity refers to the diversity and variability in the distribution of data across different devices or nodes in a FL network. In real-world scenarios, devices may possess datasets with distinct characteristics, such as varying data distributions, feature representations, and data formats. Addressing data heterogeneity is necessary to ensure effective model training and generalization across diverse datasets. In FL, data can be categorized into two main types: IID and non-IID data. IID data assumes that data samples across devices are independently and identically distributed, meaning that each device's dataset represents a random sample drawn from the same underlying distribution. Conversely, non-IID data refers to datasets where samples across devices are not identically distributed, resulting in variations in data characteristics, such as class distributions, feature correlations, and data biases.

To accommodate the diverse data distributions and facilitate model training across distributed devices, various data partitioning techniques are also employed in FL environments. These techniques include vertical partitioning, horizontal partitioning, and hybrid partitioning:

- Vertical partitioning (Liu et al., 2024) involves splitting the dataset based on features or attributes, where each device holds a subset of features for the entire dataset. This technique is suitable when devices possess complementary but non-overlapping feature sets. Vertical partitioning enables devices to collaborate on model training while preserving data privacy and confidentiality.

- Horizontal partitioning (Zhang et al., 2021) divides the dataset into subsets of samples where each device retains a portion of the data instances. With horizontal partitioning, devices have the same feature sets but distinct data samples. This approach facilitates parallelized model training across devices while preserving data locality and reducing communication overhead.

- Hybrid partitioning (Zhang et al., 2021) combines elements of both vertical and horizontal partitioning techniques, allowing for a hybrid approach to data partitioning. In hybrid partitioning, datasets may be partitioned both vertically and horizontally, accommodating diverse data structures and distributions. This flexibility enables FL systems to handle complex data heterogeneity scenarios effectively.

Our findings on this criterion are presented in Table 1. The analysis highlights several frameworks with broad coverage of the evaluated criteria, particularly PySyft, FATE, FedML, and NVFlare, which offer support for most key functionalities. However, support for vertical and hybrid data partitioning varies among them. For example, PySyft does not provide built-in capabilities for vertical or hybrid partitioning; instead, it relies on a companion library called PyVertical, requiring users to manage an additional dependency. Other frameworks, such as TFF, Flower, vantage6, Substra, and OpenFL, also demonstrate solid functionality and are actively maintained, though they currently lack support for hybrid partitioning. Ultimately, the choice between vertical and horizontal partitioning depends on the specific requirements of the application. For those seeking a versatile solution applicable to a range of scenarios, FATE, FedML, and NVFlare emerge as particularly strong candidates.

Table 1 - Data Heterogeneity and Partitioning Handling capabilities of the various frameworks considered in this analysis.

Framework	IID	non-IID	Horizontal	Vertical	Hybrid
FATE	✓	✓	✓	✓	✓
TFF	✓	✓	✓	✓	X
Flower	✓	✓	✓	✓	X
Vantage6	✓	✓	✓	X	X
FS	✓	✓	✓	✓	X
NVFlare	✓	✓	✓	✓	✓
APPLE	✓	✓	✓	✓	X
FedML	✓	✓	✓	✓	✓
OpenFL	✓	✓	✓	✓	X
Flame	✓	✓	✓	X	X
FedN	✓	✓	✓	X	X
Substra	✓	✓	✓	✓	X
PySyft	✓	✓	✓	✓	✓

Communication topologies

In FL, the network topology refers to how the various participating nodes, typically servers and clients, are arranged and how they communicate with each other. Each topology has its unique characteristics and is suitable for different scenarios and requirements. We considered 3 different topologies as follows:

- Client-Server (Yuan et al., 2024): In a client-server topology, there is one central server that coordinates the learning process. All clients (which could be devices or local systems) are connected to this central server and are not directly connected to each other. The clients train models locally on their data and send the updates—such as model gradients or weights—to the server. The server aggregates these updates, updates the global model, and then distributes the updated model back to the clients. This topology is straightforward and is the most commonly used in FL due to its simplicity in aggregating updates and managing the overall learning process.
- Decentralized (Yuan et al., 2024): A decentralized topology, also known as a peer-to-peer (P2P) network, lacks a central server that governs the learning process. Instead, clients communicate directly with one another to share model updates. This topology can enhance privacy and robustness, as there is no single point of failure or control. It also can reduce server bottlenecks. However, it may require more sophisticated protocols for synchronization, aggregation, and consistency of the model updates across the network.
- Hierarchical (Rana et al., 2023): In a hierarchical topology, there are multiple levels of nodes, with higher-level nodes serving as aggregators or coordinators for groups of lower-level nodes. This can be thought of as a tree structure, with the root being the central server, branches being intermediate nodes (which could be servers themselves), and leaves being the clients. It's a hybrid approach that combines elements of both centralized and decentralized topologies to manage large-scale deployments and reduce communication overhead. The intermediate nodes can perform some aggregation before sending updates to the central server, which can be beneficial in scenarios where clients are geographically distributed or where bandwidth is limited.

The different communication topologies of the frameworks considered are summarized in Table 2. From this overview, it is clear that four frameworks, FS, NVFlare, FedML, and Flame, support all the main communication topologies. Additionally, two other frameworks, Substra and PySyft, deserve mention for supporting at least two communication topologies each. Substra supports Client Server and Decentralized models, while PySyft offers Client Server and Hierarchical topologies. Other frameworks, such as Flower, are rapidly evolving and currently developing additional communication topologies, including hierarchical structures.

Table 2 - Communication topologies available in the frameworks considered in this analysis.

Framework	Client-Server	Decentralized	Hierarchical
FATE	✓	X	X
TFF	✓	X	X
Flower	✓	X	X
Vantage6	✓	X	X
FS	✓	✓	✓
NVFlare	✓	✓	✓
APPFL	✓	X	X
FedML	✓	✓	✓
OpenFL	✓	X	X
Flame	✓	✓	✓
FedN	✓	X	X
Substra	✓	✓	X
PySyft	✓	X	✓

Compatibility and integration

Flexibility in ML frameworks is relevant when selecting a FL framework because it enables the customization of the system to suit a wide array of use cases and data distributions. A framework that boasts flexibility is capable of supporting a multitude of ML algorithms, models, and data types, thus meeting the unique needs of diverse scenarios. Such adaptability not only broadens the framework's utility in practical applications but also ensures its competence in managing the variety and evolving characteristics of data across distributed clients. The ability to seamlessly integrate with different environments and technological stacks further amplifies a framework's effectiveness, making it a vital tool for organizations aiming to leverage FL. This versatility is critical in overcoming the challenges posed by data privacy concerns, as it allows for the development of tailored solutions that comply with regulatory standards while optimizing performance.

The analysis of the FL frameworks in this area is presented in Table 3. As shown, many frameworks are agnostic regarding the ML framework used. In some cases, such as vantage6, the developer is responsible for setting up the ML environment, while the FL communication is managed by the framework itself. Only four frameworks do not support a wide range of ML libraries: TFF, FS, APPFL, and Flame. For example, TFF is limited to TensorFlow, as it is built directly on top of that library.

Table 3 - Compatibility of each framework considered in this analysis.

Framework	Compatibility
FATE	✓
TFF	X
Flower	✓
Vantage6	✓
FS	X
NVFlare	✓
APPFL	X
FedML	✓
OpenFL	✓
Flame	X
FedN	✓
Substra	✓
PySyft	✓

Commercial usage

The adoption of FL frameworks for commercial purposes represents a major advancement in how businesses handle data privacy and implement distributed ML. For real-world applications, it is necessary that these frameworks not only possess the appropriate licenses to ensure legal compliance and protect intellectual property but also come with comprehensive documentation and a variety of tutorials (Li et al., 2022; Spinellis, 2019). Adequate licensing allows companies to deploy these technologies safely, without infringing on software patents or violating open-source agreements (Spinellis, 2019). Equally important is the availability of detailed documentation and tutorials, which are used by developers to implement and customize the FL solutions to fit specific business needs (Li et al., 2022; Spinellis, 2019).

Among the frameworks examined in this analysis, only about half are currently suitable for commercial use, as shown in Table 4. These include FATE, Flower, vantage6, NVFlare, FedN, Substra, FedML, and OpenFL. These frameworks are the most complete in terms of documentation and tutorials and are distributed under permissive licenses that allow commercial deployment. Some are already in use in real-world production settings. For example, Substra is being adopted by hospitals and biotechnology companies (Heyndrickx et al., 2022; Wang & Li, 2021). Similarly, Flower has been adopted by major companies such as Porsche, Bosch, and Samsung. On the other hand, several frameworks are not yet ready for commercial use, either due to the absence of a suitable license or because they are still in active development and lack comprehensive documentation and tutorials. A notable example is PySyft, which is still evolving and does not yet offer the level of support required for commercial applications.

Table 4 - Commercial usage of each framework considered in this analysis.

Framework	Commercial usage
FATE	✓
TFF	X
Flower	✓
Vantage6	✓
FS	X
NVFlare	✓
APPFL	X
FedML	✓
OpenFL	✓
Flame	X
FedN	✓
Substra	✓
PySyft	X

Privacy and security features

In this section, we examine the key aspects of privacy and security in FL frameworks. We discuss the distinctive features and mechanisms these frameworks employ to address such concerns, ranging from data encryption and secure multi-party computation to differential privacy and robust aggregation algorithms. Each framework offers its own approach to mitigating potential threats. In the following list, we present all the features that will be analysed:

- Secure communication: A challenge in FL is to ensure that model updates exchanged between clients and the central server are transmitted securely. To protect data during transmission, secure communication protocols such as Transport Layer Security (TLS) (Krawczyk et al., 2013) are used to encrypt information in transit, preventing unauthorized access. In addition, authentication mechanisms based on digital certificates are applied to verify the identity of the communicating parties, ensuring that model updates are exchanged only between trusted participants.
- Differential Privacy (Wei et al., 2019): Differential privacy is a technique that introduces noise into the data or model updates to limit the possibility of reverse-engineering the data and disclosing information about any individual. In FL, differential privacy can be applied during the aggregation process so that the server receives a noisy version of the model updates, which protects individual updates from being isolated and identified.

- Secure aggregation: Secure aggregation is a protocol that allows the server to compute an aggregate of model updates from multiple clients without learning any individual update. Techniques such as Secure Multi-Party Computation (SMPC) (Mugunthan et al., 2019) or Homomorphic Encryption (HE) (Park & Lim, 2022) are often used. SMPC allows for functions to be computed jointly by multiple parties without any party revealing their input to the others. HE allows computations to be performed on encrypted data without needing to decrypt it. These techniques ensure that the server gains no knowledge of the individual contributions, protecting the clients' data privacy and reducing the risk of data breaches or reconstruction attacks.

We summarize the privacy and security features of each framework in Table 5. As shown, only a few frameworks have implemented all the techniques considered in this analysis. These include Flower, NVFlare, FedML, Substra, FATE, OpenFL, and PySyft. Given the importance of privacy and security in FL, these frameworks appear to be the most suitable choices for use cases where the protection of individual user data is a primary concern.

Table 5 - Security and privacy features provided by each framework considered in this analysis.

Framework	Secure Communication	Differential Privacy	Secure Aggregation
FATE	✓	✓	✓
TFF	X	✓	✓
Flower	✓	✓	✓
Vantage6	✓	X	✓
FS	X	✓	X
NVFlare	✓	✓	✓
APPFL	✓	✓	X
FedML	✓	✓	✓
OpenFL	✓	✓	✓
Flame	✓	✓	X
FedN	✓	X	X
Substra	✓	✓	✓
PySyft	✓	✓	✓

Selected frameworks

To explore the main focus of this work, we selected two FL frameworks from those analyzed in this section. This choice was motivated by the need to avoid limiting the analysis to a single implementation and to obtain a more comprehensive understanding of their capabilities in real-world scenarios such as smart cities and edge-based environments. Flower and NVFlare were chosen due to multiple reasons. Both frameworks offer extensive and well-maintained documentation, a wide range of tutorials, and strong support for integration with common development environments, including virtual machines (VM). In addition, they demonstrate high versatility with respect to data heterogeneity and partitioning strategies, supporting both IID and non-IID data and providing horizontal partitioning mechanisms. Both frameworks are also recognized for their readiness for commercial use, offering permissive licenses and being adopted by major industry players. Finally, they include privacy and security features, such as secure communication protocols and differential privacy, making them well suited for sensitive IoT applications where the protection of user data is required.

Material and methods

In this section, we present the experimental design developed to investigate the applicability of FL in smart city contexts, particularly in scenarios involving DL and IoT. We describe four use cases based on datasets related to smart cities, aiming to evaluate the advantages and limitations of FL in supervised classification tasks. All experiments followed the same setup, using FL to train different Neural Network (NN) models, applied to three distinct datasets. The next subsections provide a detailed account of the system architecture and other experimental aspects.

Environment setup

All experiments in this study were conducted using the same development environment and operating system. The server was a desktop machine running Ubuntu 22.04, equipped with an Intel i5-13600K CPU (14 cores, 20 threads), 32 GB of DDR5 RAM, and an RTX 3060 GPU with 12 GB of memory. A Python virtual environment (version 3.10.6) was used to isolate dependencies and manage the required tools. The system included key FL components such as PyTorch 1.13.1, Flower 1.4.0, and NVFlare 2.3.0, meeting the specific requirements of the experiments.

On the client side, a realistic simulation of an FL system was created by deploying VMs as client nodes on the same physical machine used as the server. This setup enabled the exploration of varying data characteristics, processing capabilities, and communication constraints typical of real-world FL deployments. High-performance tools were used to support this architecture: Vagrant (2.3.4) orchestrated the environment, Ansible (7.5.0) handled provisioning and configuration, and VirtualBox (6.1.38) provided the virtualization layer to run multiple VMs concurrently. Each VM operated in an isolated Python virtual environment (version 3.10.6) to manage dependencies independently. OpenSSH (8.9p1) was used for secure VM management.

A single Vagrant file defined a system with multiple VMs, each allocated 3 CPU cores and 4 GB of RAM, running Ubuntu 22.04. Unique IP addresses and individual Python environments were assigned to each VM. The same FL components—PyTorch 1.13.1, Flower 1.4.0, and NVFlare 2.3.0 were used to ensure consistency across all nodes. The combined use of Vagrant and Ansible allowed for rapid deployment of clients and ensured consistency through automation, reducing configuration errors and variability.

Table 6 provides a brief summary of the testbed specifications used.

Table 6 - Hardware and software specifications for the experimental testbed.

System	Component	Name	Version
Host	Operating system	GNU/Linux Ubuntu	22.04
	ML frameworks	Pytorch	1.13.1
	FL frameworks	Flower NVFlare	1.4.0 2.3.0
	Hardware	Intel i5 13600K 32GB RAM	6+8 core DDR5
	Platform	python venv	3.10.6
	Architecture	x86	
VMs	Operating system	GNU/Linux Ubuntu	22.04
	ML frameworks	Pytorch	1.13.1
	FL frameworks	Flower NVFlare	1.4.0 2.3.0
	Hardware	Intel i5 13600K 4GB RAM	3 core DDR5
	Platform	python venv Vagrant VirtualBox Ansible	3.10.6 2.3.4 6.1.38 7.5.0

Problem domain specification

In this section, we delineate the selected domain for our FL experiments as primarily a classification problem. This determination stems from the intrinsic characteristics of the datasets employed in our research. Classification tasks are defined by the goal of categorizing input data into discrete classes, which aligns with the nature of our data, where each entry is associated with a specific, categorical outcome.

Given the classification-oriented nature of the problem domain, Neural Networks (NNs) constitute a well-suited modelling choice due to their demonstrated effectiveness in capturing complex patterns within high-dimensional data. Moreover, recent research has highlighted neural and DL approaches as enabling technologies in smart city contexts, reinforcing their relevance for data-driven applications in such domains (Jain et al., 2023; Wu et al., 2024).

To implement and train our NN models, PyTorch was selected as the primary ML tool, favored for its dynamic computation graph and extensive support for DL operations. PyTorch facilitates rapid prototyping and efficient experimentation, which are well aligned with the iterative nature of FL research.

In addition to PyTorch, secondary tools such as NumPy, pandas, and scikit-learn were integrated into our workflow to support data manipulation, preprocessing, and additional analysis tasks. NumPy offers powerful numerical computation functionalities, pandas provides high-level data structures and data manipulation tools designed for fast and easy data cleaning and analysis, and scikit-learn offers a wide range of tools for ML and statistical modelling.

Datasets

We used three datasets, all related to smart city and IoT systems. The first dataset is part of the Industrial Control System (ICS) Cyber Attack Datasets, specifically the Power System environment (Adhikari et al., 2014). This system includes four phasor measurement units (PMUs), each recording 29 features, resulting in 116 PMU-related measurement attributes. In addition, the dataset contains 12 attributes corresponding to control panel logs, Snort alerts, and relay logs associated with the PMUs. Altogether, the dataset comprises 128 input features and one target variable used for classification, which distinguishes among three possible classes.

The second dataset, known as the Extra Sensory Dataset (Vaizman et al., 2017), contains data collected from sensors embedded in smartphones and smartwatches used by 60 individuals. The dataset captures a wide range of contextual and behavioral information, with annotations covering 51 distinct activity labels that describe the users' physical and situational states. In this study, the classification task is formulated as a binary problem.

Finally, the third dataset used is the Intelligent Vehicle Perception Based on Inertial Sensing and Artificial Intelligence dataset (Menegazzo & von Wangenheim, 2020), which comprises nine sub-datasets collected using GPS, camera, inertial sensors (accelerometers and gyroscopes), a magnetometer, and a temperature sensor. As with the first dataset, the classification task involves three distinct output classes.

For all experiments, the datasets were first preprocessed to structure the data appropriately for supervised learning. Each dataset was divided into input features and target labels, and subsequently split into training, validation, and test sets following a 60/20/20 proportion. This division was performed to ensure effective model training, hyperparameter tuning, and final evaluation. After splitting, the data was formatted for compatibility with the chosen DL framework, enabling the use of data loaders to manage batch processing. During training, data shuffling was applied to prevent the model from learning unintended patterns, whereas for validation and testing, data order was preserved to ensure consistent evaluation across runs.

Models architecture

On the client side, for each of the three different used datasets, a distinct neural network architecture was designed. For the first dataset, a feedforward neural network was designed with an input dimension of 128, followed by a hidden layer of 64 units and a ReLU activation function. The final layer outputs predictions over three classes, aligning with the classification objective of the task. Also for the second dataset, the model was a NN consisting of a single linear layer with an input size of 72 and an output size of 2, indicating binary classification. The activation function used is the sigmoid function, applied to the output of the linear layer. For the last dataset, the provided model is a more complex NN consisting of three convolutional layers, each followed by a rectified linear unit (ReLU) activation, batch normalization, and dropout for regularization. After the convolutional layers, there is a global average pooling layer, and the output is then passed through two fully connected (dense) layers. The last layer has three output nodes, indicating it is designed for a classification task with three classes.

On the server side, a key component is the algorithm responsible for aggregating the locally trained models received from the clients. In all experiments, we use the Federated Averaging (FedAvg) algorithm (McMahan et al., 2023) as the mechanism for this aggregation. The server collects model updates from each client, which are the result of training on locally stored data using a common neural network architecture, and computes their average to generate a global model. FedAvg enables the integration of local learning into a unified model that reflects the diversity of data distributions across clients.

Experimental results

This section presents the experimental evaluation of FL in the context of smart city applications. Through a series of three case studies, we investigate how FL compares to centralized learning in terms of model performance and hardware resource usage, how the number of participating clients influences training outcomes, and how privacy-preserving techniques such as Homomorphic Encryption affect system efficiency. Each case study is designed to address a specific aspect of FL deployment, using the datasets introduced in the previous section. These experiments provide an analysis of the benefits and trade-offs associated with adopting FL in distributed urban environments where data privacy, scalability, and computational constraints are critical. Given that the training phase represents the most computationally demanding stage of the learning process, our evaluation emphasizes training dynamics and system-level efficiency, where the most significant differences between approaches may become evident.

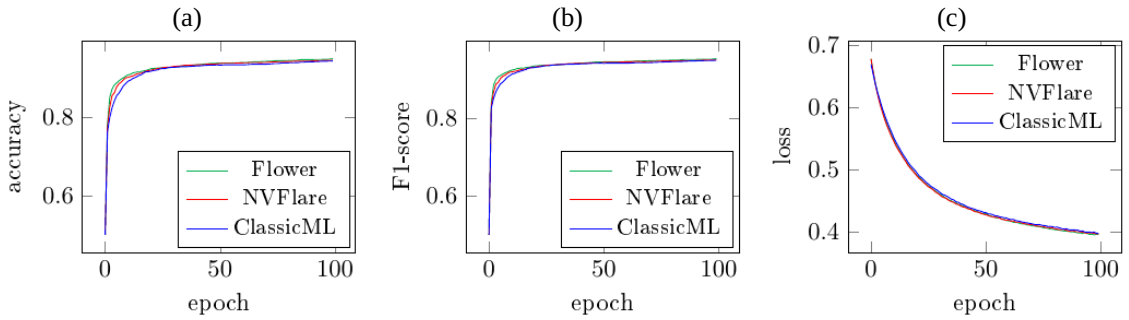
Centralized learning vs FL

In the first study case, we compare the performance of a centralized DL (referred to as classical ML) approach with two FL frameworks: NVFlare and Flower, totaling three experimental settings. The goal is to evaluate differences in model performance and hardware resource usage between centralized and federated training.

The comparison focuses on loss, accuracy, and F1-score as performance metrics. For resource analysis, we measured execution time, CPU usage, and RAM consumption on a single client. To ensure fairness, all experiments used the same model architecture, data pre-processing pipeline, and framework for training, validation, and testing. Each training scenario ran for 100 epochs. In the FL settings, we simulated centralized training by using a single client over one round. Additionally, we tested a reduced configuration with 30 epochs to assess consistency across training durations.

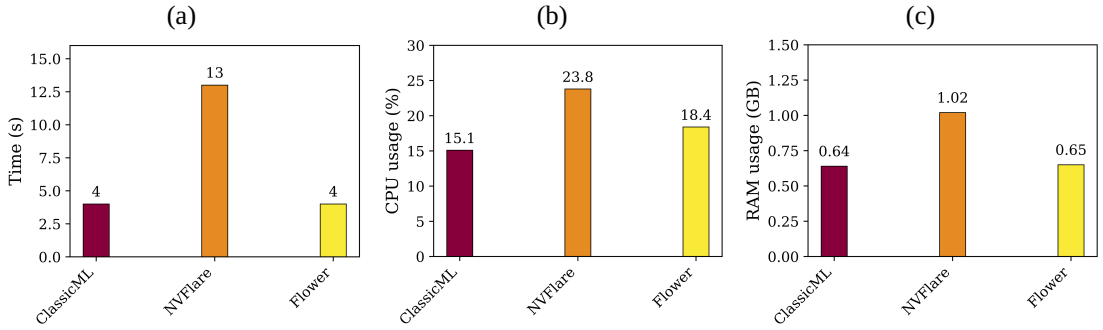
The three scenarios yield wnearly identical results. As shown in Figure 2, the training metrics demonstrate that the FL approaches maintain performance comparable to the centralized ML baseline. In addition, both FL frameworks, NVFlare and Flower, achieve similar outcomes, which indicates that they are equally effective for training in this context. The validation and test metrics follow the same trend as the training results, with all frameworks producing closely aligned performance. These findings reinforce the viability of FL for this use case and confirm that distributed training can match the effectiveness of centralized approaches.

Figure 2 - Training metrics comparison: (a) accuracy versus epoch; (b) F1-score versus epoch; and (c) loss versus epoch for the three approaches.



However, when analyzing the hardware usage on individual clients, the differences between the approaches become more pronounced. This analysis focuses on three factors: total execution time, CPU usage, and RAM consumption. As shown in Figure 3, the total execution time reflects the duration required by each framework to complete training, validation, and testing. The classical ML approach and Flower exhibit similar execution times, largely due to the simplicity of Flower's architecture. In contrast, NVFlare requires significantly more time, which can be attributed to its more complex internal design. Regarding CPU usage, displayed in the middle of Figure 3, the classical approach is the most efficient. Flower shows higher CPU consumption on average, while NVFlare exhibits the highest usage among the three. A similar trend is observed in RAM consumption, illustrated on the right side of the same Figure 3. Flower again performs comparably to the centralized ML approach, highlighting its efficiency. NVFlare, on the other hand, shows substantially higher memory usage, consistent with its architectural complexity.

Figure 3 - Training performance comparison across learning paradigms: (a) Accuracy over training epochs; (b) F1-score over training epochs; (c) Training loss evolution over training epochs. Results compare centralized machine learning (ML) with federated learning (FL) approaches using Flower and NVFlare, showing similar convergence behavior and final performance across all methods.



Node influence

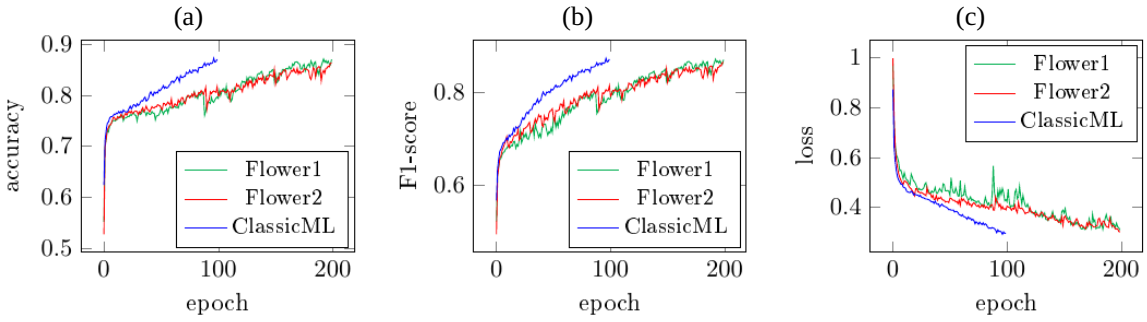
In this second case study, we evaluated the influence of individual nodes, understood as clients, in a FL setting compared to classical ML. The analysis focused on differences in hardware usage metrics between the two approaches. Given its better performance compared to NVFlare in the tests comparing centralized learning and FL, the Flower framework was selected for the experiments and tested with two distinct datasets to enhance the validity of the results. Since the ML and FL models rely on different architectures, we adopted an adjusted comparison strategy: the classical ML model was trained using all available training data, while in the FL setup, training data was distributed across clients to simulate a federated deployment scenario.

On the third dataset, to ensure a fair comparison, the centralized model was trained for 100 epochs, while the FL model was trained over 20 rounds (*num_rounds_fed*), with 10 local epochs per round (*num_epochs_fed*) and 2 clients (*num_clients_fed*). This configuration results in an equivalent number of training iterations across both setups, yielding 100 effective epochs according to equation (1):

$$central\_epochs = \frac{num\_rounds\_fed \cdot num\_epochs\_fed}{num\_clients\_fed}. \tag{1}$$

Analyzing the training metrics in Figure 4, we observe that the FL system, despite requiring significantly more epochs, is able to achieve results comparable to those obtained through the classical ML approach. This behavior arises from the fact that, in the FL scenario, each client has access to only a portion of the dataset (in this case, half) since the data is split between the two clients.

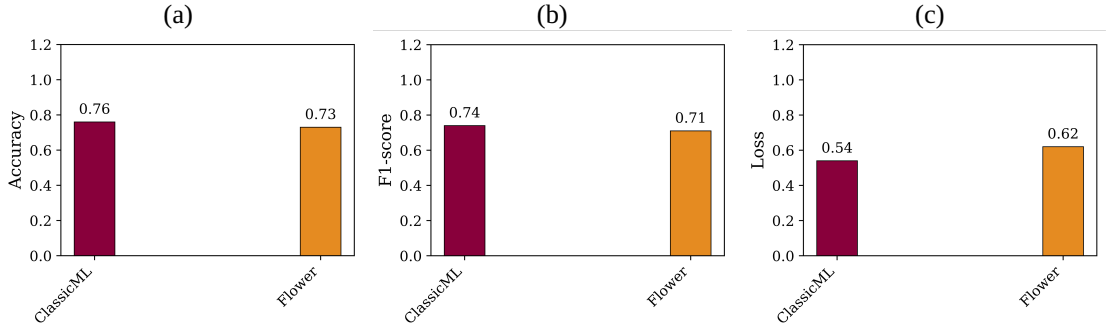
Figure 4 - Training metrics comparison for node influence analysis using the third dataset. (a) Accuracy over training epochs; (b) F1-score over training epochs; (c) Loss across training epochs. The figure compares centralized ML and FL (Flower with two clients), showing similar convergence despite data partitioning across clients.



A similar pattern is observed in the validation metrics, which closely follow the trends seen in the training results. However, when analyzing the test metrics shown in Figure 5, the FL approach performs worse than the classical ML counterpart. This discrepancy stems from the way the original dataset, composed of eight subsets, was partitioned. In the FL setup, the data was split into two groups of four subsets, allowing

each client to use two subsets for training, one for validation, and one for testing. In contrast, the classical ML model had access to the full dataset and used four subsets for training, two for validation, and two for testing. This broader data availability resulted in better internal generalization in the centralized model, which explains its superior performance in this case.

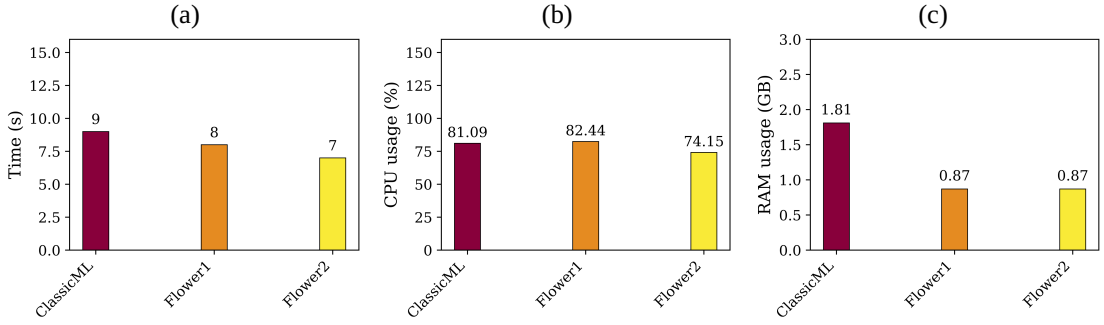
Figure 5 - Test predictive performance comparison under centralized and federated settings using the third dataset. (a) Accuracy; (b) F1-score; (c) Loss on the test set. Results indicate slightly lower generalization performance for FL due to limited data availability per client.



Turning to the hardware usage comparison, all experiments were conducted using three VMs: two were allocated for the Flower-based FL experiments, and one for the classical ML approach. To ensure a fair comparison, all VMs shared the same hardware configuration and operating system, and were provisioned using the same Vagrant file. The analysis focused on execution time, CPU usage, and RAM consumption, as in the previous case, revealing noteworthy insights. As shown in Figure 6, the classical ML experiment required more execution time than the FL experiment. This outcome is expected, as the FL setup distributes the workload across two VMs, while the classical ML runs entirely on a single VM. Despite the longer execution time, the centralized ML approach appears more efficient when considering total resource usage relative to performance.

Regarding CPU usage, Figure 6 shows that Flower consumed slightly less CPU on average than the classical ML setup, suggesting better parallel efficiency in the FL scenario. The difference becomes even more pronounced in terms of memory usage: FL maintained RAM consumption below 1 GB for both clients, whereas the classical ML approach consistently required around 2 GB. These findings further highlight the memory efficiency of the FL approach in this experimental configuration.

Figure 6 - Hardware usage comparison under centralized and federated settings using the third dataset. (a) Execution time (seconds) across centralized ML and two Flower clients; (b) CPU utilization (%) across centralized ML and two Flower clients; (c) RAM consumption (GB) across centralized ML and two Flower clients. Results demonstrate improved memory efficiency and distributed workload in FL



Following the same methodology, we conducted an additional experiment using the Extra Sensory dataset (second dataset). Table 7 presents an overview of the experimental configurations adopted in the two scenarios considered in the node influence analysis. In this case, we adopted an FL configuration with 5 clients instead of 2, which required a greater number of training rounds and local epochs to ensure a fair comparison with the classical ML baseline.

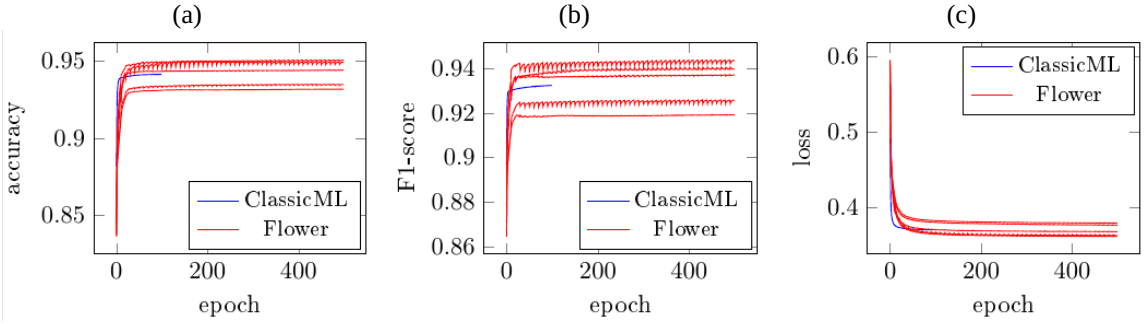
Table 7 - Experimental configuration for node influence analysis.

Dataset	Approach	# Clients	Rounds	Local Epochs	# VMs	Data Split Strategy
Third Dataset	Classical ML	–	–	–	1	Full dataset available centrally; 4 subsets for training, 2 for validation, 2 for testing
Third Dataset	FL (Flower)	2	20	10	2	Dataset divided into 2 groups of subsets; per client: 2 training, 1 validation, 1 testing
Second Dataset	Classical ML	–	–	–	1	Full dataset used centrally
Second Dataset	FL (Flower)	5	50	10	5	Data distributed across 5 clients

Specifically, the FL model was trained over 50 rounds, with each client performing 10 local epochs per round, totaling 500 local epochs across the system. Based on equation (1), this configuration yields the same number of effective iterations as the centralized ML model trained for 100 epochs, thus maintaining consistency in the comparative analysis.

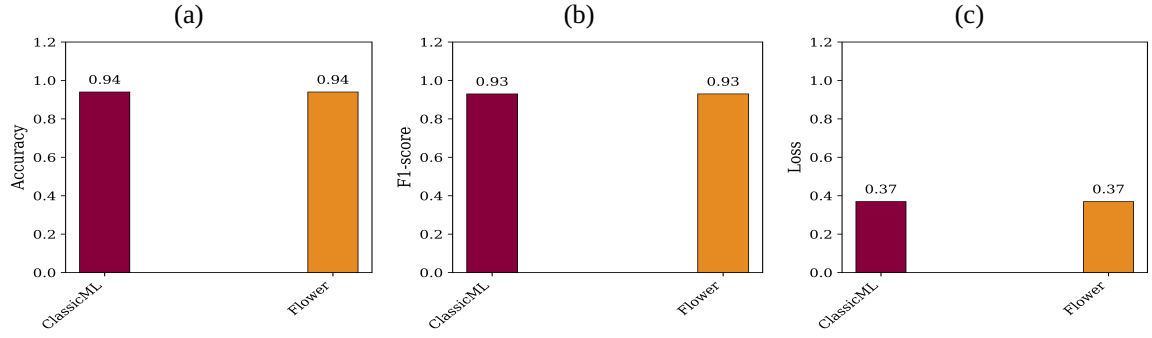
The training metrics shown in Figure 7 highlight how the five Flower clients yield markedly different results, due to the heterogeneous nature of their local datasets. This divergence underscores the gap between the performance of classical ML and FL in this context. However, since training occurs on data local to each client, more meaningful insights emerge when analyzing the validation and testing metrics on the server side.

Figure 7 - Training metrics comparison between Classic ML (centralized) and a 5-client FL setup using the second dataset. (a) Accuracy over training epochs; (b) F1-score over training epochs; (c) Loss over training epochs. The variability across clients highlights the impact of data heterogeneity in federated environments.



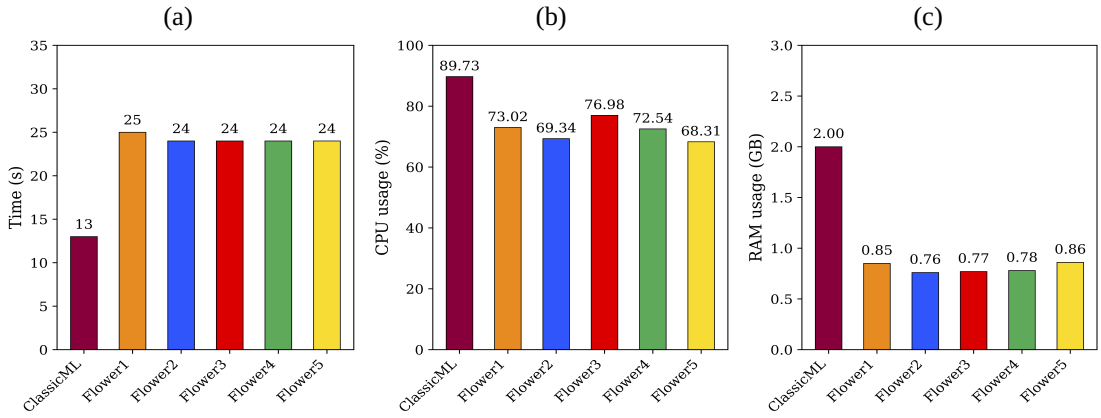
As shown in Figure 8, the validation results are similar to the classical ML approach, demonstrating the efficiency of the FL technique in an IoT system such as the one employed here. Unlike the training phase, where results varied significantly across clients, the validation phase benefits from client collaboration. This cooperation allows the server to aggregate client-side outcomes at the end of each round, effectively producing an average that leads to consistent and reliable results. A similar conclusion can be drawn from the testing phase, where the outcomes closely match those of the classical ML approach.

Figure 8 - Validation performance comparison between Classic ML (centralized) and a 5-client FL setup using the second dataset. (a) Accuracy; (b) F1-score; (c) Loss. Aggregated FL results closely match centralized performance, demonstrating the effectiveness of collaborative model updates.



In this scenario, five VMs were used, one for each Flower client, reflecting the increased number of participants in the FL setting. The results obtained for hardware usage are even more revealing than in the previous case. As shown in Figure 9, the total execution time for each Flower client exceeds that of the classical ML approach. This outcome is expected, since the FL setup requires more epochs to achieve comparable performance, which in turn increases the overall execution time. It is also important to note that while FL uses five VMs, the ML experiment runs on a single VM, further emphasizing the relative time efficiency of the centralized ML approach.

Figure 9 - Hardware usage comparison between Classic ML (centralized) and a 5-client FL setup using the second dataset. (a) Execution time (seconds) per client; (b) CPU utilization (%); (c) RAM consumption (GB). Results emphasize lower per-device resource usage in FL, despite increased overall execution time.



When analyzing CPU utilization, also shown in Figure 9, we observe that the average CPU usage for each Flower client is approximately 70%, whereas the ML process maintains a consistent 100% utilization. This indicates that FL is more CPU-efficient, suggesting that it can be executed on less powerful machines, such as embedded systems, without compromising functionality. This characteristic is particularly relevant in IoT environments where high-performance hardware may not be available.

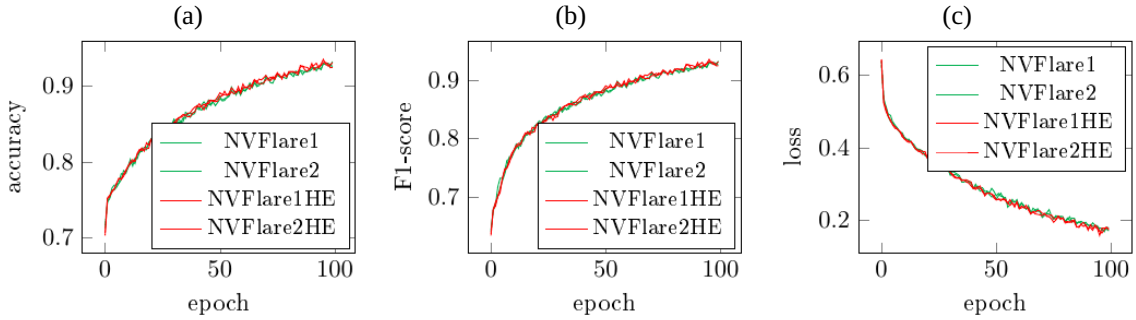
Despite this advantage, due to the distributed architecture, FL still takes longer to reach equivalent results, as discussed earlier. RAM usage provides additional evidence of the efficiency of the FL approach. While the classical ML setup consistently consumes over 2 gigabytes of RAM, each Flower client averages less than 0.8 gigabytes. This significant difference, also visible in Figure 9, reinforces the suitability of FL for environments with limited resources. It demonstrates that in smart city applications, where data is distributed and local devices have restricted capabilities, FL offers a practical alternative to centralized training.

Data privacy and security

In this third case study, we analyze the impact of applying data privacy and security techniques within a FL context. Privacy and security are among the core motivations behind the development of FL, as it is designed to enable collaborative learning without exposing raw data. To explore this aspect, we focus on one of the most widely used and effective methods in this domain: Homomorphic Encryption (HE). For our experiments, we employed the NVFlare framework, which is the only one among the two frameworks considered that supports this technique. The experiment uses Dataset 3 and compares FL performance with and without the use of HE. Specifically, we examine variations in hardware usage and model performance metrics across both configurations. Two settings were evaluated: one with 2 clients and another with 5 clients. Each configuration was executed for 5 epochs and 20 rounds, both in the HE-enabled scenario and in the baseline FL setup without HE.

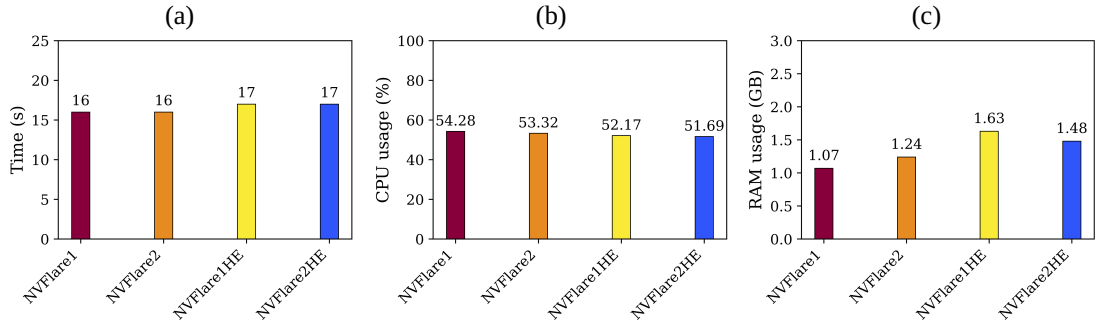
Starting with the first configuration, which uses 2 clients, the training metrics shown in Figure 10 indicate that the performance of the HE experiment is very similar to that of the non-HE version. Although Homomorphic Encryption introduces overhead due to the added encryption of model updates sent to the server, this does not significantly impact performance. This stability is partly explained by the relatively low complexity of the dataset used: an attribute common to many datasets in IoT and smart city applications.

Figure 10 - Training performance with and without Homomorphic Encryption (HE) in a 2-client FL setup. (a) Accuracy; (b) F1-score; (c) Loss across training rounds. The figure shows that enabling HE has minimal impact on model convergence and predictive performance.



Looking more closely at hardware resource usage, Figure 11 shows a slight difference in execution time between the two experiments. This increase is possibly due to the overhead introduced by the encryption and decryption processes in the HE-enabled setup. Given that not all IoT and smart city applications require strong data privacy and security, it may be more efficient to forgo HE in scenarios where runtime performance is a priority.

Figure 11 - Hardware usage comparison between 2-client FL setups with and without Homomorphic Encryption (HE): (a) Execution time (seconds); (b) CPU utilization (%); (c) RAM consumption (GB). HE increases runtime and memory usage while maintaining similar CPU utilization.

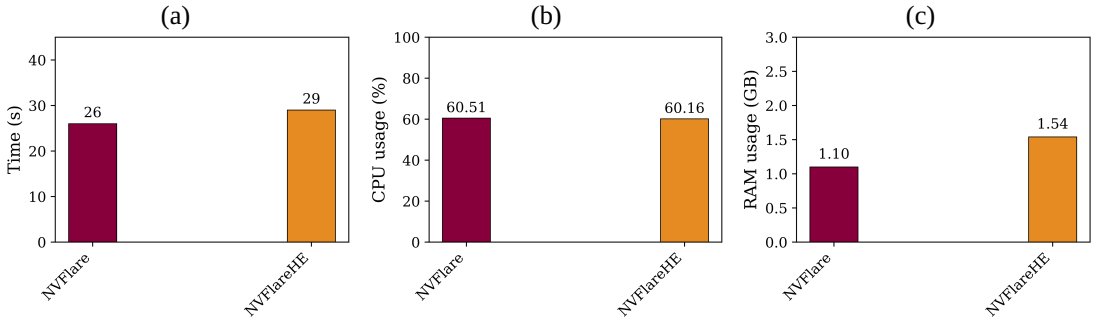


If we examine CPU usage in Figure 11, it remains the same for both scenarios (HE and non-HE). However, since the experiment without HE runs for a shorter duration, it is overall more efficient in terms of CPU time. The third and final parameter to consider is RAM usage, also shown in Figure 11. In this case, the HE approach requires more memory, averaging around 1.6 GB, while the standard NVFlare setup uses less, approximately 1.1 GB. This further illustrates that although HE enhances data privacy and security, it also increases the demand on system resources. Therefore, it may not be the most suitable option in environments where protecting user data is not a primary concern.

Examining the results from the second configuration with five clients, the training, validation, and test metrics are very similar between the scenario without Homomorphic Encryption and the one using this technique. More interesting insights emerge when we examine hardware usage. As shown in Figure 12, there is a difference of about 10% in execution time between the two experiments. This increase is caused by the overhead introduced by data encryption and decryption when using Homomorphic Encryption. These findings are consistent with those from the experiment involving only two clients and demonstrate that the HE approach is not ideal when time efficiency is a priority.

Also in this case, as shown in Figure 12, the situation is similar to what we observed with the two-client configuration. Both experiments show approximately the same average CPU usage. However, since the experiment without Homomorphic Encryption requires less runtime, it performs better overall in terms of CPU efficiency. Regarding RAM usage, also illustrated in Figure 12, the HE approach demands more memory on average compared to the non-HE setup. These results closely match those from the two-client experiment, with the only difference being a slightly higher peak in RAM usage for both configurations. This consistency across different client settings highlights that while Homomorphic Encryption improves data privacy and security, it also increases resource consumption. Therefore, it may not be the best option in environments where protecting user data is not the highest priority.

Figure 12 - Hardware usage comparison between 5-client FL setups with and without Homorphic Encryption (HE): (a) Execution time (seconds); (b) CPU utilization (%); (c) RAM consumption (GB). Results confirm that HE increases computational overhead, particularly in execution time and memory usage.



Conclusion and future work

Until recently, the literature on FL was limited both in the number of frameworks available and in comparing this new distributed approach with the classic centralized approach of ML. In this work, we investigated FL through a two-step approach. First, we conducted a comparative analysis of several FL frameworks, focusing on their features, architecture, usability, and support for various ML paradigms. This initial evaluation allowed us to identify the most suitable frameworks for further experimentation. In the second phase, we selected two representative FL frameworks and compared their performance against a traditional centralized ML approach. Our experiments demonstrated that while FL frameworks introduce some overhead and complexity compared to classic ML, they provide a valuable trade-off between data privacy and performance. Among the FL frameworks tested, each exhibited specific strengths depending on the context and use case. Future work will focus on expanding the scope of our evaluation to include additional frameworks and more complex datasets. Furthermore, we plan to investigate optimization techniques for FL, such as client selection strategies and asynchronous training, to improve scalability and adaptability in real-world deployments. In addition, we will make interpretability a central direction for future work by integrating explainable AI (XAI) methods to enhance transparency, support debugging, and build trust in FL-enabled decision-making.

Author Contributions

L. di Lauro: Conceptualization, Data Curation, Formal Analysis, Investigation, Methodology, Programs, Validation, Visualization, Writing. **B. B. Zarpelão**: Conceptualization, Methodology, Project Management, Supervision, Validation, Visualization, Writing. **S. Barbon Junior**: Conceptualization, Formal Analysis, Methodology, Project Management, Supervision, Validation, Visualization, Writing.

Conflicts of Interest

The authors declare no conflict of interest.

References

Adhikari, U., Pan, S., Morris, T., Borges, R., & Beaver, J. (2014). *Industrial control system (ICS) cyber attack datasets: Power system datasets*. https://sites.google.com/a/uah.edu/tommy-morris-uah/ics-data-sets

Al-Huthaifi, R., Li, T., Huang, W., Gu, J., & Li, C. (2023). Federated learning in smart cities: Privacy and security survey. *Information Sciences*, *632*, 833–857. https://doi.org/10.1016/j.ins.2023.03.033

Alla, K. R., & Thangarasu, G. (2023). Federated learning for IoT devices in smart cities: A particle swarm optimization-based approach. In Institute of Electrical and Electronics Engineers, *International Conference SmartTechCon* [Proceedings]. Second International Conference on Smart Technologies for Smart Nation, Singapore, Singapore, 730–734. https://doi.org/10.1109/SmartTechCon57526.2023.10391420

Annas, G. J. (2003). HIPAA regulations—a new era of medical-record privacy? *New England Journal of Medicine*, *348*(15), 1486–1490. https://doi.org/10.1056/NEJMlim035027

Arafeh, M., Hammoud, A., Otrok, H., Mourad, A., Talhi, C., & Dziong, Z. (2022). Independent and identically distributed (IID) data assessment in federated learning. In Institute of Electrical and Electronics Engineers, *GLOBECOM 2022* [Proceedings]. Global Communications Conference, Rio de Janeiro, Brazil, 293–298. https://doi.org/10.1109/GLOBECOM48099.2022.10001718

Banabilah, S., Aloqaily, M., Alsayed, E., Malik, N., & Jararweh, Y. (2022). Federated learning review: Fundamentals, enabling technologies, and future applications. *Information Processing & Management*, *59*(6), 103061.

Beutel, D. J., Topal, T., Mathur, A., Qiu, X., Fernandez-Marques, J., Gao, Y., Sani, L., Li, K. H., Parcollet, T., de Gusmão, P. P. B., & Lane, N. D. (2022). Flower: A friendly federated learning research framework, [preprint]. *Arxiv*, 1–15. https://arxiv.org/abs/2007.14390

Blinder, A., & Perlroth, N. (2018). A Cyberattack Hobbles Atlanta, and Security Experts Shudder. *The New York Times*, 27. https://www.nytimes.com/2018/03/27/us/cyberattack-atlanta-ransomware.html

Cho, H., Mathur, A., & Kawsar, F. (2022). Flame: Federated learning across multi-device environments. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*. *6*(3), 1–29. https://doi.org/10.1145/355028

Cui, L., Xie, G., Qu, Y., Gao, L., & Yang, Y. (2018). Security and privacy in smart cities: Challenges and opportunities. *IEEE Access*, *6*, 46134–46145. https://doi.org/10.1109/ACCESS.2018.2853985

Djenouri, Y., Michalak, T. P., & Lin, J. C. (2023). Federated deep learning for smart city edge-based applications. *Future Generation Computer Systems*, *147*, 350–359. https://doi.org/10.1016/j.future.2023.04.034

Ekmefjord, M., Ait-Mlouk, A., Alawadi, S., Åkesson, M., Singh, P., Spjuth, O., Toor, S., & Hellander, A. (2022). Scalable federated machine learning with FEDn. [Preprint]. *Arxiv*, 1-15. https://arxiv.org/abs/2103.00148

Foley, P., Sheller, M. J., Edwards, B., Pati, S., Riviera, W., Sharma, M., Moorthy, P. N., Wang, S., Martin, J., Mirhaji, P., Shah, P., & Bakas, S. (2022). OpenFL: The open federated learning library. *Physics in Medicine & Biology*, *67*(21), 214001. https://doi.org/214001.10.1088/1361-6560/ac97d9

Galtier, M. N., & Marini, C. (2019). Substra: a framework for privacy-preserving, traceable and collaborative Machine Learning. [Preprint]. *Arxiv*, 1–22. https://arxiv.org/abs/1910.11567

Georgiou, D., & Lambrinoudakis, C. (2021). Data protection impact assessment (DPIA) for cloud-based health organizations. *Future Internet*, *13*(3), 66. https://doi.org/10.3390/fi13030066

Gracias, J. S., Parnell, G. S., Specking, E., Pohl, E. A., & Buchanan, R. (2023). Smart cities—a structured literature review. *Smart Cities*, *6*(4), 1719–1743. https://doi.org/10.3390/smartcities6040080

Hasan, J. (2023). Security and Privacy Issues of Federated Learning. [Preprint]. *Arxiv*, 1-6. https://arxiv.org/abs/2307.12181

He, C., Li, S., So, J., Zeng, X., Zhang, M., Wang, H., Wang, X., Vepakomma, P., Singh, A., Qiu, H., Zhu, X., Wang, J., Shen, L., Zhao, P., Kang, Y., Liu, Y., Raskar, R., Yang, Q., Annavaram, M., & Avestimehr, S. (2020). FedML: A research library and benchmark for federated machine learning [Preprint]. *Arxiv*, 1-18. https://arxiv.org/abs/2007.13518

Heidari, A., Navimipour, N. J., & Unal, M. (2022). Applications of ML/DL in the Management of Smart Cities and Societies: A Systematic Literature Review. *Sustainable Cities and Society*, *85*, 104089. https://doi.org/10.1016/j.scs.2022.104089

Heyndrickx, W., Mervin, L., Morawietz, T., Sturm, N., Friedrich, L., Zalewski, A., Pentina, A., Humbeck, L., Oldenhof, M., Niwayama, R., Schmidtke, P., Fechner, N., Simm, J., Arany, A., Drizard, N., Jabal, R., Afanasyeva, A., Loeb, R., Verma, S., & Ceulemans, H. (2022). MELLODDY: cross pharma federated learning at unprecedented scale unlocks benefits in QSAR without compromising proprietary information [Preprint]. *Chemrxiv*, 1-23. https://doi.org/10.26434/chemrxiv-2022-ntd3r

Jain, A., Gue, I. H., & Jain, P. (2023). Research trends, themes, and insights on artificial neural networks for smart cities towards SDG-11. *Journal of Cleaner Production*, *412*, 137300. https://doi.org/10.1016/j.jclepro.2023.137300

Jiang, J. C., Kantarci, B., Oktug, S., & Soyata, T. (2020). Federated learning in smart city sensing: Challenges and opportunities. *Sensors*, *20*(21), 6230. https://doi.org/10.3390/s20216230

Karimireddy, S. P., Veeraragavan, N. R., Elvatun, S., & Nygård, J. F. (2023). Federated Learning Showdown: The Comparative Analysis of Federated Learning Frameworks. In Institute of Electrical and Electronics Engineers, *FMEC 2023* [Proceedings]. Eighth International Conference on Fog and Mobile Edge Computing. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=10305961

Konečný, J., McMahan, B., & Ramage, D. (2015). Federated Optimization:Distributed Optimization Beyond the Datacenter datacenter [Preprint]. *Arxiv*, 1-5. https://arxiv.org/abs/1511.03575

Krawczyk, H., Paterson, K. G., & Wee, H. (2013). On the Security of the TLS Protocol: A Systematic Analysis. In R. Canetti & J. A. Garay (Eds.), *Advances in Cryptology – CRYPTO 2013* (pp. 429–448). Springer. https://doi.org/10.1007/978-3-642-40041-4_24

Kuang, Z., & Chen, C. (2023). Research on smart city data encryption and communication efficiency improvement under federated learning framework. *Egyptian Informatics Journal*, *24*(2), 217–227. https://doi.org/10.1016/j.eij.2023.02.005

Li, X., Moreschini, S., Zhang, Z., & Taibi, D. (2022). Exploring factors and metrics to select open source software components for integration: An empirical study. *Journal of Systems and Software*, *188*, 111255. https://doi.org/10.1016/j.jss.2022.111255

Liu, Y., Fan, T., Chen, T., Xu, Q., & Yang, Q. (2021). FATE: An industrial grade platform for collaborative learning with data protection. *Journal of Machine Learning Research*, *22*, 1–6. https://doi.org/https://www.jmlr.org/papers/v22/20-815.html

Liu, Y., Kang, Y., Zou, T., Pu, Y., He, Y., Ye, X., Ouyang, Y., Zhang, Y.-Q., & Yang, Q. (2024). Vertical federated learning: Concepts, advances, and challenges. *IEEE Transactions on Knowledge and Data Engineering*, *36*(7), 3615–3634. https://doi.org/10.1109/tkde.2024.3352628

Majeed, U., Khan, L. U., Yaqoob, I., Kazmi, S. M. A., Salah, K., & Hong, C. S. (2021). Blockchain for IoT-based smart cities: Recent advances, requirements, and future challenges. *Journal of Network and Computer Applications*, *181*, 103007. https://doi.org/10.1016/j.jnca.2021.103007

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., & y Arcas, B. A. (2023). Communication-Efficient Learning of Deep Networks from Decentralized Data [Preprint]. *Arxiv*, 1-11. https://arxiv.org/abs/1602.05629

Menegazzo, J., & von Wangenheim, A. (2020). Multi-Contextual and Multi-Aspect Analysis for Road Surface Type Classification Through Inertial Sensors and Deep Learning. In Institute of Electrical and Electronics Engineers, *Brazilian Symposium on Computing Systems Engineering (SBESC)* [Proceedings]. 10º Brazilian Symposium on Computing Systems Engineering (SBESC), Florianopolis, Brazil. https://doi.org/10.1109/SBESC51047.2020.9277846

Moncada-Torres, A., Martin, F., Sieswerda, M., Soest, J., & Geleijnse, G. (2021). Vantage6: An open source privacy preserving federated learning infrastructure for secure insight exchange. *AMIA*

Annual Symposium Proceedings Archive, *2020*, 870–877. https://pmc.ncbi.nlm.nih.gov/articles/PMC8075508/

Mugunthan, V., Polychroniadou, A., Byrd, D., & Balch, T. H. (2019). SMPAI: Secure multi-party computation for federated learning. In *Annual Conference on Neural Information Processing Systems* [Proceedings]. 33rd Annual Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada. https://www.jpmorgan.com/content/dam/jpm/cib/complex/content/technology/ai-research-publications/pdf-9.pdf

Nguyen, H., Nawara, D., & Kashef, R. (2024). Connecting the indispensable roles of IoT and artificial intelligence in smart cities: A survey. *Journal of Information and Intelligence*, *2*(3), 261–285. https://doi.org/10.1016/j.jiixd.2024.01.003

Park, J., & Lim, H. (2022). Privacy-preserving federated learning using homomorphic encryption. *Applied Sciences*, *12*(2), 734. https://doi.org/10.3390/app12020734

Rana, O., Spyridopoulos, T., Hudson, N., Baughman, M., Chard, K., Foster, I., & Khan, A. (2023). Hierarchical and Decentralised Federated Learning. [Preprint]. *Arxiv*, 1-11. https://arxiv.org/abs/2304.14982

Riedel, P., Schick, L., von Schwerin, R., Reichert, M., Schaudt, D., & Hafner, A. (2024). Comparative analysis of open-source federated learning frameworks – a literature-based survey and review. *International Journal of Machine Learning and Cybernetics*, *15*, 5257–5278. https://doi.org/10.1007/s13042-024-02234-z

Roth, H. R., Cheng, Y., Wen, Y., Yang, I., Xu, Z., Hsieh, Y.-T., Kersten, K., Harouni, A., Zhao, C., Lu, K., Zhang, Z., Li, W., Myronenko, A., Yang, D., Yang, S., Rieke, N., Quraini, A., Chen, C., Xu, D., … Feng, A. (2022). NVIDIA FLARE: Federated Learning from Simulation to Real-World. [Preprint]. *Arxiv*, 1-13. https://arxiv.org/abs/2210.13291

Ryu, M., Kim, Y., Kim, K., & Madduri, R. K. (2022). APPFL: Open-source software framework for privacy-preserving federated learning. In Institute of Electrical and Electronics Engineers, *IPDPSW 2022* [Proceedings]. International Parallel and Distributed Processing Symposium Workshops, Lyon, France. https://doi.org/10.1109/IPDPSW55747.2022.00175

Spinellis, D. (2019). How to select open source components. *Computer*, *42*(12), 103–106. https://doi.org/10.1109/MC.2019.2940809

The European Parliament and the Council of the European Union. (2016). *Regulation (EU) 2016/679*. http://data.europa.eu/eli/reg/2016/679/oj

Ullah, A., Anwar, S. M., Li, J., Nadeem, L., Mahmood, T., Rehman, A., & Saba, T. (2024). Smart cities: The role of internet of things and machine learning in realizing a data-centric smart environment. *Complex & Intelligent Systems*, *10*(1), 1607–1637. https://doi.org/10.1007/s40747-023-01175-4

Vaizman, Y., Ellis, K., & Lanckriet, G. (2017). Recognizing detailed human context in the wild from smartphones and smartwatches. *IEEE Pervasive Computing*, *16*(4), 62–74. https://doi.org/10.1109/MPRV.2017.3971131

Valente, R., Senna, C., Rito, P., & Sargento, S. (2023). Federated learning framework to decentralize mobility forecasting in smart cities. In Institute of Electrical and Electronics Engineers, *NOMS 2023-2023* [Proceedings]. NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium, Miami, FL, USA, 1–5. https://doi.org/10.1109/NOMS56928.2023.10154456

Wang, B., Li, H., Guo, Y., & Wang, J. (2023). Ppflhe: A privacy-preserving federated learning scheme with homomorphic encryption for healthcare data. *Applied Soft Computing*, *146*, 110677.

Wang, B., & Li, Z. (2021). Healthchain: A privacy protection system for medical data based on blockchain. *Future Internet*, *13*(10), 247. https://doi.org/10.3390/fi13100247

Wei, K., Li, J., Ding, M., Ma, C., Yang, H. H., Farhad, F., Jin, S., Quek, T. Q. S., & Poor, H. V. (2019). Federated Learning with Differential Privacy: Algorithms and Performance Analysis [Preprint]. *Arxiv*, 1-15. https://arxiv.org/abs/1911.00222

Wu, P., Zhang, Z., Peng, X., & Wang, R. (2024). Deep learning solutions for smart city challenges in urban development. *Scientific Reports*, *14*(1), 5176. https://doi.org/10.1038/s41598-024-55928-3

Xie, Y., Wang, Z., Gao, D., Chen, D., Yao, L., Kuang, W., Li, Y., Ding, B., & Zhou, J. (2022). FederatedScope: A flexible federated learning platform for heterogeneity. *Proceedings of the VLDB Endowment*, *16*(5), 1059–1072. https://doi.org/0.14778/3579075.3579081

Yuan, L., Wang, Z., Sun, L., Yu, P. S., & Brinton, C. G. (2024). Decentralized federated learning: A survey and perspective. *IEEE Internet of Things Journal*, *11*(21), 34617–34638. https://doi.org/10.1109/JIOT.2024.3407584

Zhang, X., Yin, W., Hong, M., & Chen, T. (2021). Hybrid Federated Learning: Algorithms and Implementation. [Preprint]. *Arxiv*, 1-8. https://arxiv.org/abs/2012.12420

Zhu, H., Xu, J., Liu, S., & Jin, Y. (2021). Federated learning on non-IID data: A survey [Preprint]. *Arxiv*, 1-29. https://arxiv.org/abs/2106.06843

Ziller, A., Trask, A., Lopardo, A., Szymkow, B., Wagner, B., Bluemke, E., Nounahon, J. M., Passerat-Palmbach, J., Prakash, K., Rose, N., Ryffel, T., Reza, Z. N., & Kaissis, G. (2021). PySyft: A library for easy federated learning. In Rehman, M.H., Gaber, M.M. (Eds), *Federated Learning Systems. Studies in Computational Intelligence* (Vol 965, pp. 111-139). Springer. https://doi.org/10.1007/978-3-030-70604-3_5