

Federated Learning with TensorFlow Federated for Soil Classification in Smart Agriculture

Marwen Ghabi^{1*}, Sofiane khalfallah², Hela Ltifi³

National School of Electronics and Telecommunications of Sfax, Tunisia¹
University of Sfax, Research Groups in Intelligent Machines, Sfax, Tunisia^{2,3}

Abstract—In the field of smart agriculture, the use of agricultural data plays a key role in optimizing practices and improving decision-making. However, the centralization of this data raises significant challenges in terms of confidentiality and protection of information from farms. To address these challenges, Federated Learning (FL), implemented with TensorFlow Federated (TFF), allows multiple participants to collaborate on training a global model while keeping their data locally. This study presents an innovative solution combining federated learning with TensorFlow Federated and a modular microservices architecture to deploy predictive models using a Machine Learning as a Service (MLaaS) approach. The proposed architecture relies on the TensorFlow ecosystem to orchestrate distributed training, local model aggregation, and the deployment of inference services. It includes components dedicated to data management, local training, federated coordination, and the provision of application programming interfaces (APIs) to deliver predictions for agricultural applications. This approach aims to facilitate the integration, maintenance, and scalability of artificial intelligence services in distributed environments. A case study on soil texture classification is conducted to evaluate the performance of the proposed approach and the effectiveness of the federated learning process. The experimental results demonstrate the potential of this architecture for developing distributed artificial intelligence services in smart agriculture while limiting the sharing of raw agricultural data among participants.

Keywords—Federated learning; tensorflow federated; artificial intelligence; smart agriculture; Machine Learning as a Service (MLaaS); microservices architecture; soil classification; data privacy; predictive models; scalability; resilience

I. INTRODUCTION

In a global context where efficiency and sustainability have become agricultural priorities, artificial intelligence (AI) and its sub-domains such as machine learning (ML) offer promising solutions to address these challenges. Artificial intelligence (AI) can significantly improve crop management, resource optimization and crop yield forecasting. However, the collection and processing of massive data from farms around the world raises major privacy and data security concerns.

Federated learning (FL) appears as an effective solution to this problem, allowing us to train AI models without centralizing sensitive data. This decentralized approach ensures that information remains on local devices (e.g., ground sensors or weather stations) and only model parameters such as gradients are shared between devices to create a global model. Therefore, this significantly reduces the risks of privacy breaches while maintaining high model performance.

In the area of smart agriculture, FL is particularly relevant. By combining diverse data sources such as ground sensors, drones, or satellite images, AI models can adapt to specific local conditions while benefiting from shared global knowledge. This approach improves the accuracy of agricultural forecasts and optimizes agricultural practices by integrating local data while preserving confidentiality [1]. In addition, the deployment of machine learning as a service (MLaaS) within a microservices architecture allows agricultural actors to easily access AI models via standardized interfaces, without the need for in-depth technical expertise. This creates a flexible and scalable infrastructure for farms, allowing them to adapt these technologies to their specific needs. MLaaS architecture is particularly suitable for distributed environments such as the Internet of Things, where devices are geographically dispersed but require centralized access to AI services [2].

Thus, the combination of FL and MLaaS within a microservices infrastructure not only protects agricultural data but also accelerates the adoption of AI technologies in this critical sector. Demonstrate that this approach provides effective scalability and facilitates the integration of AI solutions into farmers' daily operations.

II. LITERATURE REVIEW

State-of-the-art Federated learning has become a key approach to overcoming privacy challenges in training artificial intelligence models. Traditionally, the data was centralized in a single server to drive ML models, which posed serious confidentiality issues, particularly in sensitive areas such as health and agriculture. FL allows data to be kept on local devices while sharing only model updates. This approach, well explored in various fields, is just beginning to develop in the smart agriculture sector [1] [2].

Federated learning allows participating clients to retain their raw training data locally while exchanging model parameters with a coordinating server. However, this approach does not by itself provide a formal guarantee against information leakage through model updates. The privacy properties of the proposed implementation should therefore be interpreted in terms of local data retention rather than as a formal privacy guarantee.

In agriculture, sensors, satellite images and drones generate vast amounts of data that must be analyzed to make informed decisions about crops, soil management, and weather. However, the geographic and environmental diversity of farms makes it difficult to train a single model that could be applied to all three regions (Mediterranean region, Saharan region,

*Corresponding author.

continental region). FL allows us to circumvent this obstacle by driving models specific to each region while benefiting from a federated aggregation of information. For example, federated learning (FL) can be shown to improve the accuracy of agricultural recommendations by integrating local data without requiring explicit sharing [1]. At the same time, the adoption of Machine Learning as a Service (MLaaS) in agriculture allows for democratizing access to sophisticated models without requiring end-users (farmers, and farm managers) to have advanced technical knowledge [3] [4]. The MLaaS uses a micro-services architecture, where AI services are accessible via APIs and can be easily integrated with existing farm management tools. This also allows for better model scalability and maintenance, as well as reduced infrastructure costs [6].

Identified gaps:

- Limited integration of federated learning for soil classification: Although federated learning has attracted increasing attention in distributed machine learning, its application to soil texture classification in smart agriculture remains relatively underexplored.
- Limited investigation of MLaaS for complex agricultural machine learning tasks: Existing research on Machine Learning as a Service (MLaaS) in agriculture has not sufficiently addressed the deployment and provision of complex machine learning models for soil classification.
- Lack of integrated architectural approaches: The combined use of federated learning, MLaaS, and microservices for soil texture classification has received limited attention, particularly in architectures designed to support distributed model training and scalable model serving.

This study investigates the integration of federated TensorFlow training and local KNN classification within an MLaaS-oriented architecture for soil texture classification. The contribution is evaluated in terms of model training, local classification, and software architecture. The novelty of this combination is discussed in relation to the existing literature.

III. CURRENT LIMITATIONS AND CHALLENGES

Although federated learning (FL) offers several advantages, several challenges continue to limit its large-scale adoption in the agricultural sector. One of the main challenges concerns communication between local devices and the central server, particularly in rural areas where network connectivity may be limited. Furthermore, the repeated exchange of model updates can incur significant communication costs, potentially limiting the practicality of federated learning in agricultural environments [2], [9]. Data heterogeneity also poses a challenge for agricultural applications based on the Internet of Things (IoT). Agricultural devices can produce data of variable quality, according to different formats and frequencies of acquisition. These characteristics must be taken into account when designing machine learning services adapted to IoT environments [9].

In summary, federated learning and Machine Learning as a Service (MLaaS) offer promising opportunities for smart

agriculture. However, their large-scale adoption requires addressing connectivity constraints, improving the interoperability of IoT devices, and ensuring model robustness in distributed environments. These aspects represent important research directions for developing more reliable agricultural decision-support systems that can better accommodate real-world farming conditions.

IV. METHODOLOGY

This section describes the methodology for combining federated training of a TensorFlow neural network with local K-Nearest Neighbors (KNN) classification within a Machine Learning as a Service (MLaaS) architecture. The Flower framework coordinates federated training using the Federated Averaging (FedAvg) strategy. While the neural network parameters are aggregated by the central server, each client independently trains its KNN classifier using local training data. This implementation separates federated neural network training from local KNN classification.

By integrating KNN classification into the proposed Machine Learning as a Service (MLaaS) architecture and leveraging the TensorFlow ecosystem for federated neural network training, we aim to:

1) *Optimizing soil classification:* Combine federated training of a TensorFlow neural network with local KNN classification to evaluate their applicability to soil texture classification.

2) *Keep training data local:* Allow each participating client to retain its local training data while exchanging neural network parameters through the Flower framework using the FedAvg strategy. However, federated learning alone does not guarantee complete data privacy.

3) *Facilitate model access via microservices:* Design a modular microservices architecture to provide access to soil classification functionality through REST APIs. This architecture supports modularity and maintainability and can facilitate service replication to accommodate increasing workloads when supported by the deployment infrastructure.

4) *Improve distributed training and resource management:* Use federated learning to coordinate neural network training across participating clients while aggregating model parameters through FedAvg. Keep KNN classifiers independent at each client. The approach aims to support distributed model training, while reductions in storage costs, communication overhead, and computational costs require further experimental evaluation.

5) *Support agricultural innovation and decision-making:* Integrate TensorFlow-based federated learning and local KNN classification into the MLaaS architecture to facilitate soil texture classification and provide information that can support agricultural decision-making in smart agriculture.

A. Data Collection and Preparation

Soil texture classification is an important task in smart agriculture because soil properties influence agricultural management and plant growth. Soil texture is commonly characterized by the relative proportions of sand, silt, and clay, which are

used to determine soil texture classes according to the USDA soil classification system [7].

This study uses the Ebergötzen soil dataset to develop and evaluate the proposed classification models [16]. The dataset includes four input features: sand, silt, clay, and soil depth. The target soil texture classes are derived from the sand, silt, and clay percentages using predefined classification rules based on the USDA soil texture classification system [7].

Before training, the dataset is prepared by selecting the input features and generating the corresponding class labels. Feature standardization is applied using parameters calculated from the training data only to avoid data leakage. The training data are then distributed among participating clients. Each client trains a TensorFlow neural network locally, while the Flower framework coordinates model-parameter aggregation using the Federated Averaging (FedAvg) strategy [11]. In parallel, each client trains its own K-Nearest Neighbors (KNN) classifier independently. Both approaches are evaluated using the designated test dataset to assess their soil texture classification performance.

B. Federated Learning and Local KNN Classification

The proposed approach combines federated learning of a neural network with independent local K-Nearest Neighbors (KNN) classification for soil texture prediction. The federated learning process is implemented using TensorFlow Federated (TFF) and the weighted Federated Averaging (FedAvg) strategy [11]. Two participating clients contribute to the training process, with each client using its local soil data to train the neural network. At each communication round, the TFF learning process aggregates the model updates according to the weighted FedAvg strategy to update the global neural network. The updated global model is then used in subsequent training rounds. The raw training samples are not explicitly transferred to the central aggregation process.

In parallel, each client independently trains a local KNN classifier using its own training data. The classifier predicts soil texture classes by identifying the nearest training samples according to a distance metric and applying distance-based weighting to their class labels. Unlike the neural network, the local KNN classifiers are not included in the FedAvg aggregation process. Each client evaluates its own KNN classifier on the validation dataset, and the resulting performance metrics are reported separately. Thus, the architecture combines federated neural network training with independent local KNN classification, without constructing a global KNN model.

Although raw training samples remain local to the clients during federated training, this does not guarantee complete privacy, since model updates may potentially reveal information about local data [5].

Fig. 1 presents the following three components of the architecture:

1) *Data collection service*: This microservice is responsible for collecting data from various sources such as IoT sensors or images captured by drones. The data is stored in a distributed manner across different local nodes to ensure low latency and better management of local resources. In

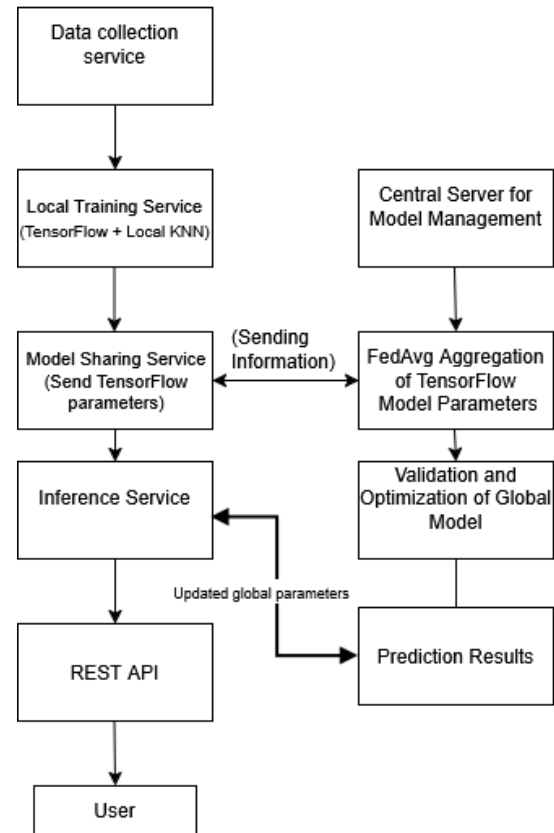


Fig. 1. Proposed architecture for federated tensorflow training with local KNN classification.

smart farming, this data may include parameters such as soil temperature, water content, or images of the terrain.

2) *Model management service*: The central server is the core of the federated learning architecture (Fig. 2). Implemented with TensorFlow Federated (TFF), it orchestrates communication between the different local nodes and ensures model lifecycle management. Its main role is to collect the locally updated TensorFlow model parameters transmitted by each node, to aggregate them using the Federated Averaging (FedAvg) strategy, and to produce a global and optimized version of the neural network model. The server also manages the coordination of training iterations, continuous validation of overall performance, and distribution of updates to participating nodes. Finally, it integrates model management mechanisms (backup, versioning, TensorFlow checkpoints) that ensure the traceability, robustness and reproducibility of the federated process, while respecting security and data confidentiality constraints.

3) *Inference service (tensorflow runtime)*: The service exposes REST APIs allowing users, such as farmers, to access the trained global federated neural network model and obtain recommendations on crops or resource management (e.g. irrigation). Input data (such as field characteristics) is processed by the service, which uses the global federated model, complemented by each node's local KNN classifier, to generate and return accurate and reliable predictions.

C. Deployment via MLaaS

The global model obtained is then deployed via a micro-services architecture as a machine learning service. Each service corresponds to a specific function of the ML pipeline, including data management, coaching, and inference. Services are encapsulated in Docker containers and orchestrated using Kubernetes to ensure the scalability and resilience of the system [2].

Users, such as farmers or farm managers, can interact with the system via RESTful APIs to get predictions on specific soil types or agricultural recommendations.

This deployment also allows for a continuous update of the global model, as new data sets collected can be integrated into the federated drive process without interrupting service operations.

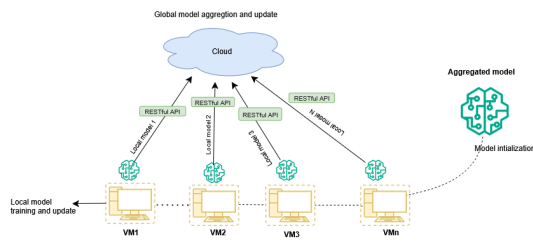


Fig. 2. Federated learning architecture: Cloud model.

V. RESULTS AND DISCUSSION

A. Data Preparation

Soil texture is an essential classification feature, used both in the field and in the laboratory, to define the different classes of soils according to their physical properties. It can be assessed by qualitative methods, such as tactile examination, or by quantitative approaches, such as the hydrometric method. Soil texture plays a major role in agriculture, notably for assessing the suitability of land for crops and predicting its behaviour in relation to environmental conditions and management practices, such as drought or limestone amendment requirements.

This classification is based on the analysis of soil particles less than two millimetres, including sand, silt and clay. International systems, such as the USDA (United States Department of Agriculture) soil taxonomy and the World Reference Base (WRB) for Soil Resources, distinguish 12 textural classes, while the UK's Agricultural Decision Support Systems (ADAS) identifies 11. These categories are established according to the relative proportions of sand, silt and clay quoted [7]. Soil texture directly influences its properties, thus determining plant growth, notably through water retention capacity, permeability and ease of work. In the United States, the USDA distinguishes twelve major textural classes: sand, silty sand, sandy loam, silt, very fine silt, silty clay, sandy clay, silty-sandy clay and clay [8]. To simplify this classification, the use of the textural triangle of soils is commonly preferred, as illustrated below.

Reading the soil texture triangle illustrated in Fig. 3:

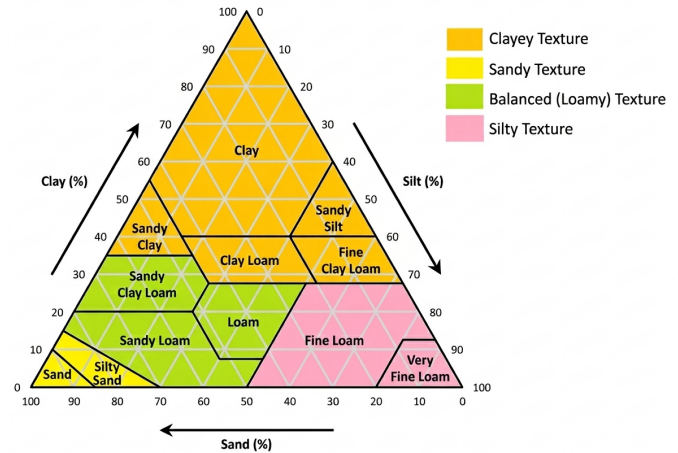


Fig. 3. Triangular diagram of soil textural classes based on USDA particle size.

- Step 1: Locate, at the bottom of the triangle, the number corresponding to the percentage of sand present in the soil sample.
- Step 2: Identify, along the right side of the triangle, the number representing the percentage of silt in the soil.
- Step 3: Determine the intersection point of the two previous lines.

Classification of soil textures (Table I):

- The table below summarizes the reading of the soil texture triangle and facilitates its interpretation.
- It presents the percentages of the different soil components allowing to obtain a textural classification.

TABLE I. USDA TEXT CLASSES

Soil types	Sand	Silt	Clay	Class textural
Sandy soils (coarse texture)	86-100	0-14	0-10	Sand
	70-86	0-30	0-15	Silty sand
Silty soils (texture moderately coarse)	50-70	0-50	0-20	Sandy loam
Silty soils (texture average)	23-52	28-50	7-27	Silt
	20-50	74-88	0-27	Silty silt
	0-20	88-100	0-12	Very fine silt
Silty soils (texture moderately fine)	20-45	15-52	27-40	Clay silt
	45-80	0-28	20-35	Clayey sandy silt
	0-20	40-73	27-40	Silty clay silt
Clay soils (fine texture)	40-65	0-20	35-55	Sandy clay
	0-20	40-60	40-60	Silty clay
	0-45	0-40	40-100	Clay

B. Acquire Data

In this study, we use the dataset Ebergötzen soil dataset, available on the Kaggle platform. Unlike a synthetic database built from the soil texture triangle, this dataset provides real observations of soil composition [16].

It includes 2,660 instances for training and 550 instances for testing, each instance being described by the percentages of sand, silt and clay, as well as soil depth.

The data were collected in the region of Ebergötzen, in central Germany, over an area of about 10 km × 10 km, and

cover several depth levels (0–10 cm, 10–30 cm, 30–50 cm, 50–70 cm, and 70–90 cm).

For the experiments conducted in this study, data were pre-processed and soil texture classes were determined according to the classification criteria adopted in our methodology. The processed version of the dataset used in experiments, as well as the resources associated with pre-processing, are publicly available in a GitHub repository [17]. This dataset thus constitutes a real experimental basis for evaluating the performance of soil texture classification in a distributed learning context.

C. Retrieving the Dataset

The CSV (Comma-Separated Values) format is widely used for the representation of data sets in machine learning, due to its simplicity and compatibility with many analysis tools.

In this study, the dataset is provided as CSV files containing the variables describing the soil composition.

Each line of the file corresponds to a soil sample and groups the following attributes:

- sample identifier (ID).
- the percentage of sand.
- the percentage of silt.
- the clay percentage.
- the depth of the soil.
- textural class (label).

In a CSV file, the header describes information about each field. Two main ways of organizing this heading can thus be distinguished:

- Case I: When the data file has a header, column names are automatically assigned from it.
- Case II: When the data file does not contain a header, it is necessary to manually assign a name to each column.

Fig. 4 provides an accurate representation of the distribution of input attributes using box diagrams. These visualizations make it possible to summarize the main statistical characteristics of the variables (sand, silt and clay), in particular the median, the quartiles, the range of values as well as the dispersion of the data, and facilitate the identification of any outliers.

It is also possible to create a histogram for each input variable in order to analyze its distribution. The histogram (Fig. 5) is a graphical representation of data in the form of bars of different heights, where the values are grouped by intervals. This visualization allows to observe the frequency of values, analyze the shape of the distribution (symmetry or asymmetry) and highlight the concentration of data in certain intervals.

We can now examine the interactions between variables using the dispersion matrix (pair plot) presented in Fig. 6. This figure combines two types of complementary visualizations: On the diagonal, histograms illustrate the individual distribution of each variable (sand, silt and clay), allowing

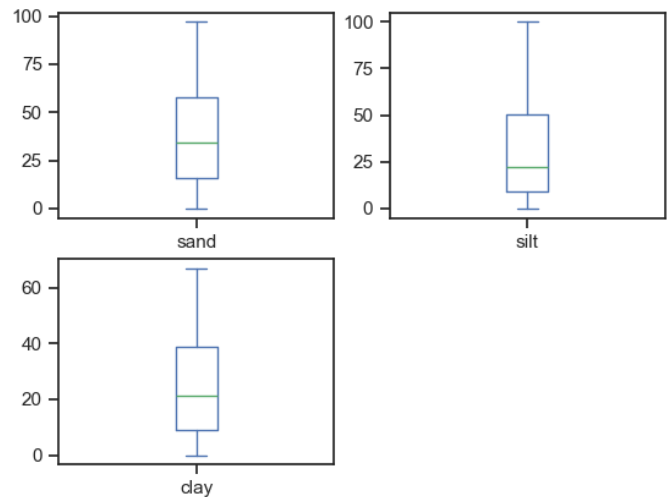


Fig. 4. Visualization of the distribution of input variables of the data set using box plots.

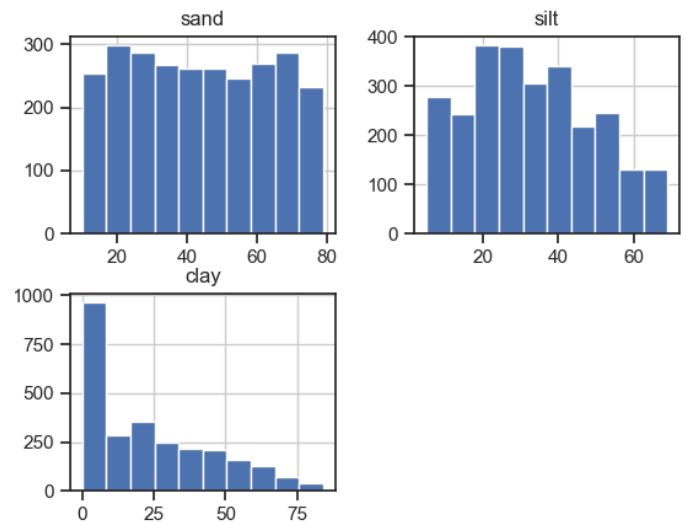


Fig. 5. Distribution of input variables represented by histograms.

to observe the distribution of values, their dispersion and possible asymmetries. The relationships between each pair of variables are represented by off-diagonal scatter plots. Scatter plots allow identifying structural relationships between input variables. We observe notably diagonal alignments and groupings of points, which indicate the existence of linear or quasi-linear relationships between certain pairs of variables. These structures show a significant correlation, suggesting that some variables are not independent. For example, the relationships between sand, silt and clay show opposite or complementary trends, which is consistent with the nature of soil composition data, where an increase in one component can be accompanied by a decrease in another. In addition, the dispersion of points around observed trends makes it possible to evaluate the strength of these relationships: a narrow concentration along one direction indicates a strong correlation, while a wider dispersion reflects a weaker relationship. This visualization is employed to explore the dependencies between variables,

detect potential correlations, and gain a better understanding of the overall data structure before the modeling phase.

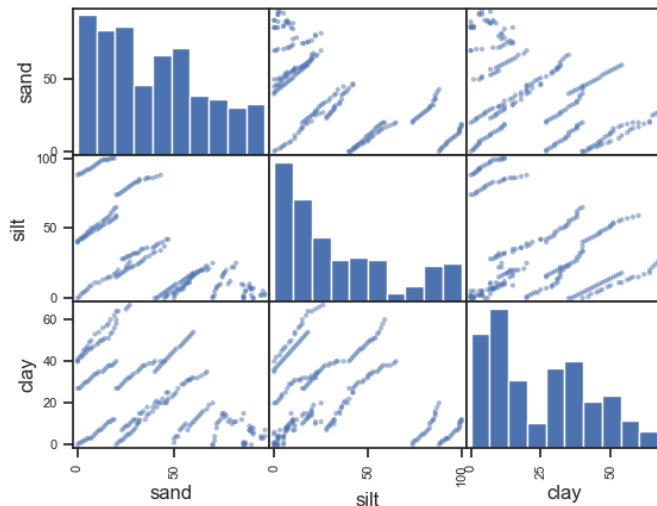


Fig. 6. Diffusion matrix diagram for each input variable for the set data.

D. Soil Classification With Federated Learning and the KNN Algorithm

The K-Nearest Neighbors (KNN) classifier achieved an accuracy of approximately 97% on the evaluated dataset, as shown in Fig. 7. The classification report provides precision, recall, F1-score, and support for each soil texture class, enabling a more detailed assessment of the classifier's performance. A REST API can be used to view the AI model report using the KNN algorithm. The classification report shows, for each class, metrics such as accuracy, recall, F1-score and support. It allows to evaluate the performance of the model, provided that the validation data set correctly reflects the studied population. This accuracy is slightly higher than traditional centralized models, especially for soils with complex geophysical characteristics [1] [2].

In the proposed implementation, KNN is evaluated independently on each participating client using its local training data. Unlike the federated neural network, the KNN classifiers do not participate in the FedAvg aggregation process. Their performance is assessed separately using classification metrics, including accuracy, precision, recall, and F1-score. Comparing the performance of the federated model with that of a conventional centralized model, we observed that the Federated algorithm provided comparable or even superior results without requiring the transfer of sensitive data between three devices. Results are consistent with recent studies that show Federated algorithm is effective in heterogeneous and dispersed data environments [3] [4].

Fig. 7 presents the KNN classification report obtained on the evaluation dataset. The classifier achieved an accuracy of approximately 97% on 532 samples. However, the results vary across classes, and the classes with zero support cannot be reliably evaluated using this dataset. Therefore, the overall accuracy should be interpreted alongside the class-level precision, recall, F1-score, and support values.

```
KNN Accuracy: 0.9699248120300752
[[115  1  0  0  0  0  0  0  0  0  0  1  0]
 [ 2 32  0  0  0  0  1  0  0  0  0  0  0]
 [ 0  0 38  0  0  0  0  3  0  0  0  0  0]
 [ 0  0  0 47  0  0  0  1  0  0  0  0  0]
 [ 0  0  0  0 40  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 48  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 34  1  0  0  0  0  0]
 [ 0  0  0  2  0  0  1 126  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0 40  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0 95  0  1  1]
 [ 0  0  0  0  0  0  0  0  0  0 10  1  1]
 [ 0  0  0  0  0  0  0  0  0  0  1  0 11]]
```

Classification Report (KNN):				
	precision	recall	f1-score	support
clay	0.98	0.98	0.98	117
clay loam	0.97	0.91	0.94	35
loam	1.00	0.93	0.96	41
loamy sand	0.96	0.98	0.97	48
sand	0.00	0.00	0.00	0
sandy clay	1.00	1.00	1.00	8
sandy clay loam	0.94	0.97	0.96	35
sandy loam	0.96	0.98	0.97	129
silt	0.00	0.00	0.00	0
silt loam	0.99	0.99	0.99	96
silty clay	0.91	0.91	0.91	11
silty clay loam	0.85	0.92	0.88	12
accuracy			0.97	532
macro avg	0.80	0.80	0.80	532
weighted avg	0.97	0.97	0.97	532

Fig. 7. KNN classification report.

E. Performance and Convergence of the Federated Learning Model

The federated learning process was implemented using TensorFlow Federated (TFF) and the weighted Federated Averaging (FedAvg) strategy. Two clients participated in the federated learning process, with each client locally building the model on its own subset of soil data. At each communication cycle, the parameters updated locally by the clients are transmitted to the central server, where they are aggregated using the FedAvg strategy and then redistributed to the participating clients.

Fig. 8 shows the evolution of the distributed loss function over 10 rounds of federated communication. This evolution enables the analysis of the convergence behaviour of the TensorFlow model in a distributed learning environment.

To further assess the stability of the federated training process, a second execution was performed using the same experimental configuration. Fig. 9 presents the evolution of the distributed training loss during this second execution.

Fig. 8 illustrates the evolution of the distributed training loss over ten communication rounds. The loss decreases overall from approximately 0.6343 in the first round to 0.1700 in the tenth round. Some fluctuations occur during training, and the minimum loss of approximately 0.1559 is reached in the ninth round. These results indicate an overall reduction in the training objective, although the loss does not decrease

```
INFO : [SUMMARY]
INFO : Run finished 10 round(s) in 5.42s
INFO : History (loss, distributed):
INFO : round 1: 0.6343335509300232
INFO : round 2: 0.43211209774017334
INFO : round 3: 0.34359508752822876
INFO : round 4: 0.28331050276756287
INFO : round 5: 0.23901790380477905
INFO : round 6: 0.258925199508667
INFO : round 7: 0.19097110629081726
INFO : round 8: 0.1805529147386551
INFO : round 9: 0.15585988759994507
INFO : round 10: 0.16998815536499023
```

Fig. 8. Evolution of the global federated training loss over 10 communication rounds.

monotonically at every round.

The total execution time for the ten communication rounds was approximately 5.42 seconds in the first run. The overall reduction in the loss indicates that the TensorFlow model progressively improves its performance during the training process through the successive aggregation of its parameters using the Federated Averaging (FedAvg) strategy.

It is important to clarify that the loss values presented in Fig. 8 correspond to the TensorFlow neural network model trained in the federated setting, and not to the K-Nearest Neighbors (KNN) classifier. In the proposed implementation, KNN is trained locally at each client using its local training data, and its performance is evaluated separately using classification metrics, including precision, recall, and F1-score. Unlike the TensorFlow neural network, KNN does not have trainable parameters such as weights or gradients and is therefore not aggregated using the FedAvg strategy.

Accordingly, Fig. 8 should be interpreted as an indicator of the convergence of the federated TensorFlow model, while the performance of the KNN classifier is analyzed separately based on its classification results.

```
INFO : [SUMMARY]
INFO : Run finished 10 round(s) in 5.36s
INFO : History (loss, distributed):
INFO : round 1: 0.6440250277519226
INFO : round 2: 0.44043874740600586
INFO : round 3: 0.3662516474723816
INFO : round 4: 0.2813313603401184
INFO : round 5: 0.24352940917015076
INFO : round 6: 0.2125033438205719
INFO : round 7: 0.18548183143138885
INFO : round 8: 0.17580284178256989
INFO : round 9: 0.1692625880241394
INFO : round 10: 0.14895257353782654
```

Fig. 9. Evolution of the distributed training loss during a second federated execution.

Fig. 9 presents the distributed training loss during a second execution. The loss decreases overall from approximately 0.6440 in the first round to 0.1490 in the tenth round. The

total execution time is approximately 5.36 seconds, which is close to the 5.42 seconds recorded during the first run. Unlike the first execution, the loss decreases continuously across the ten communication rounds, without a noticeable increase in the final round. The training process was completed in approximately 5.36 s, which is close to the 5.42 s recorded during the first execution.

Both executions show a similar overall reduction in training loss, with final values of approximately 0.1700 and 0.1490, respectively. These observations suggest a consistent training trend under the tested configuration. However, additional runs and experiments with different client data distributions are required to establish the robustness and reproducibility of the training process.

===== VALIDATION =====

=== VALIDATION RESULTS ===
Loss: 0.1805
Accuracy: 0.9323

=== CONFUSION MATRIX TF ===

```
[[105  5  0  0  0  4  0  0  0  0  3  0]
 [  0 30  2  0  0  0  0  0  0  0  0  2]
 [  0  1 40  0  0  0  0  0  0  0  0  0]
 [  0  0  0 53  0  0  0  0  0  0  0  0]
 [  0  0  0  0  0  0  0  0  0  0  0  0]
 [  0  0  0  0  0 13  0  0  0  0  0  0]
 [  0  0  0  0  0  1 34  0  0  0  0  0]
 [  0  0  6  7  0  0  2 111  0  0  0  0]
 [  0  0  0  0  0  0  0  0  0  0  0  0]
 [  0  0  1  0  0  0  0  0  0  89  0  0]
 [  0  0  0  0  0  0  0  0  0  0  6 11]
 [  0  0  0  0  0  0  0  0  0  0  1 15]]
```

=== CONFUSION MATRIX KNN ===

```
[[113  2  0  0  0  2  0  0  0  0  0  0]
 [  1 30  3  0  0  0  0  0  0  0  0  0]
 [  0  0 36  0  0  0  0  0  0  5  0  0]
 [  0  0  0 52  0  0  0  1  0  0  0  0]
 [  0  0  0  0  0  0  0  0  0  0  0  0]
 [  0  0  0  0  0 10  3  0  0  0  0  0]
 [  0  1  0  0  0  0 32  2  0  0  0  0]
 [  0  0  0  1  0  0  4 121  0  0  0  0]
 [  0  0  0  0  0  0  0  0  0  0  0  0]
 [  0  1  0  0  0  0  0  0  0  89  0  0]
 [  2  0  0  0  0  0  0  0  0  0  5  0]
 [  0  3  0  0  0  0  0  0  0  0  0 13]]
```

=== KNN classification report ===

	precision	recall	f1-score	support
clay	0.97	0.97	0.97	117
clay loam	0.81	0.88	0.85	34
loam	0.92	0.88	0.90	41
loamy sand	0.98	0.98	0.98	53
sand	0.00	0.00	0.00	0
sandy clay	0.83	0.77	0.80	13
sandy clay loam	0.82	0.91	0.86	35
sandy loam	0.98	0.96	0.97	126
silt	0.00	0.00	0.00	0
silt loam	0.95	0.99	0.97	90
silty clay	1.00	0.71	0.83	7
silty clay loam	1.00	0.81	0.90	16
accuracy			0.94	532
macro avg	0.77	0.74	0.75	532
weighted avg	0.94	0.94	0.94	532

Fig. 10. KNN classification report on the validation set.

Fig. 10 presents the confusion matrices of the federated neural network and the KNN classifier on the evaluation dataset. The detailed classification report shown in the figure corresponds to KNN and indicates an overall accuracy of approximately 94%. A direct comparison between the two approaches requires evaluating both models on the same validation samples and reporting the corresponding classification metrics.

The federated neural network is trained using TensorFlow

Federated (TFF) and the weighted Federated Averaging (FedAvg) strategy, whereas KNN is evaluated separately as a local baseline and does not participate in the federated aggregation process. The results show an overall reduction in training loss over ten communication rounds in both experimental runs.

Overall, these experiments illustrate the feasibility of federated neural network training for soil texture classification under the tested configuration. However, further experiments are needed to compare the federated neural network with KNN and a centralized baseline under the same evaluation conditions and to assess performance across different client data distributions.

F. Evaluation and Validation

The model underwent a rigorous evaluation phase on an independent test dataset, carefully selected to reflect a wide variety of agricultural conditions (soil types, geographical areas and varied physico-chemical characteristics). This approach ensures that the performances obtained are not biased by training data and that they reflect well the ability of the model to generalize to new contexts.

To measure the quality of classification, several performance indicators were used, notably:

- The accuracy, which measures the overall proportion of correct predictions.
- The recall, which evaluates the model's ability to correctly identify positive samples.
- And the F1 score, which is a balanced compromise between precision and recall.

In addition, the evaluation focused on the specific impact of federated learning. This has been compared to a traditional centralized approach, where all data is gathered in one server. The results show that federated learning achieves comparable performance while strengthening the protection of farmers' privacy, since no raw data is shared. Thus, this approach reconciles classification efficiency and compliance with ethical and regulatory constraints related to data confidentiality [3] [9].

G. MLaaS System Performance and Optimization

The microservices-based architecture allows to decompose the Machine Learning as a Service (MLaaS) system into several functional components that can be developed, deployed and updated independently. This organization is adapted to the proposed framework, in which the various components of the system, including data collection, model management, and inference services, must operate in a distributed manner while allowing the retention of training data at the participating nodes [6] [2].

In order to facilitate the deployment and maintenance of system components, a Continuous Integration/Continuous Deployment (CI/CD) pipeline has been integrated into the architecture. Fig. 11 shows the main steps of the process, from managing source code to deploying containerized services.

- Visual Studio Code (VS Code) is used as a development environment to implement and maintain the various software components of the system, including

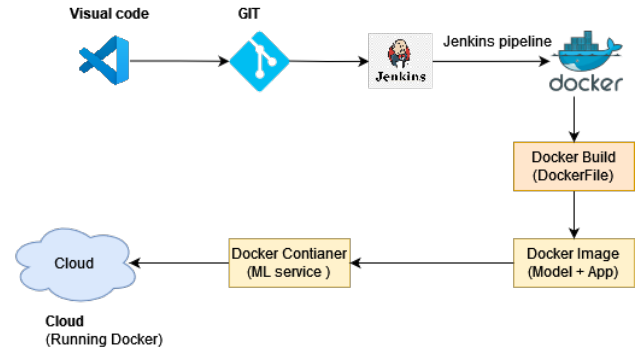


Fig. 11. Deployment architecture for MLaaS in federated learning.

MLaaS services, federated learning scripts, and necessary deployment files.

- Git is used for managing and versioning source code, Docker files, and system-related configurations. The Git repository is also the source from which the CI/CD pipeline retrieves new versions of the project.
- Jenkins ensures the automation of the CI/CD pipeline. After retrieving the project from the Git repository, Jenkins executes the various steps necessary to build, verify, and prepare software components for deployment.

The various components of the system are then encapsulated in Docker images. This containerization allows for the isolation of services and provides a repeatable execution environment. Each component of the MLaaS can thus be packaged with its dependencies and deployed independently. The generated images are executed as Docker containers on the target infrastructure, which facilitates the management and maintenance of the different services.

When multiple containers need to be deployed and managed in a distributed infrastructure, Kubernetes can be used as the orchestration layer. It allows you to manage the deployment and execution of containers as well as the coordination of different departments. This organization notably facilitates the separation of system components and their management in a distributed environment.

Thus, the DevOps pipeline constitutes a complementary operating layer to the federated learning framework. Federated Learning is responsible for distributed learning from the data stored locally by participants, while containerization, orchestration and CI/CD mechanisms facilitate deployment, the update and maintenance of the MLaaS software components. This separation allows to preserve the independence between the local data used for learning and the software components deployed in the infrastructure.

The integration of these mechanisms thus provides a more modular and reproducible deployment environment for MLaaS services. It allows the software components of the architecture to be upgraded without directly modifying the training data stored at the participating nodes.

H. Comparison with Existing Approaches

One of the main contributions of this study is the application of federated learning to smart agriculture, an area that has yet to be explored so far. Unlike centralized drive models, our approach ensures that local data remains protected while allowing for global analysis across multiple farm devices and regions [3]. Compared to previous work using MLaaS models for the Internet of Things (IoT), our approach has been distinguished by better data privacy management and improved performance in geographically disparate environments [9] [10].

The results also show that federated learning can outperform traditional data management methods in environments with limited network connections, as is often the case in rural areas. This opens the way for wider adoption of this technology, not only in agriculture but also in other sectors where data privacy and limited access to infrastructure are major concerns [11] [12].

I. Limitations of the Approach and Prospects for Improvement

Although the results of this study are promising, some limitations remain. One of the main limitations is the latency of communications between local devices and the central server, which can affect the speed of federated model training. This is particularly problematic in agricultural areas that are poorly connected. In addition, the heterogeneity of IoT devices and data formats can lead to inconsistencies in local model updates [13].

In the future, more research is needed to improve the model's resilience to these challenges. One possible way would be to explore approaches for compression of data before transmission, as well as advanced optimization techniques to reduce the bandwidth required when exchanging between local devices and the central server. In addition, the integration of reinforcement algorithms could allow for a dynamic adaptation of the model to specific local conditions, thus further improving the accuracy of predictions [14] [15].

VI. CONCLUSIONS AND FUTURE WORK

Federated learning is an innovative and promising solution for agricultural data management in the context of smart agriculture. Leveraging TensorFlow Federated (TFF), this approach offers a robust alternative to centralized methods, allowing the training of AI models without massive data transfer, while ensuring local information confidentiality and security. In this study, we have shown that the use of TFF, combined with the KNN algorithm for soil classification, allows to obtain high precision results while adapting to the geographical and technological constraints of farms.

The integration of our solution into a microservices architecture via a MLaaS service has demonstrated its effectiveness in terms of scalability, flexibility and speed of deployment. Thanks to TensorFlow, the orchestration of distributed training, federated aggregation and inference becomes seamless, providing a smooth interaction between different system users and the possibility for continuous updates. This technology lays the way for a transformation of agricultural practices, by facilitating decision-making and optimizing crop management through personalized recommendations based on soil data.

Nevertheless, some limitations persist, notably communication latency and the heterogeneity of IoT devices, which require further research. Areas for improvement include the optimization of federated coordination processes with TensorFlow, network resource management in environments with limited connectivity, as well as the integration of advanced techniques such as reinforcement learning or model and data compression to reduce computational load.

In conclusion, the federated learning with TensorFlow, integrated into a distributed services architecture, represents a major advance for smart agriculture. It combines high performance, data protection and scalability, thus constituting an essential lever for developing a more sustainable, innovative agriculture that respects the confidentiality of agricultural data.

ACKNOWLEDGMENT

The authors would like to thank the individuals who provided guidance and valuable feedback during the development of this work. Their contributions helped improve the quality of the present paper

DECLARATIONS

Competing Interests

The authors declare that they have no conflict of interest, financial or otherwise, likely to have influenced the work presented in this article. The authors have no personal, professional, or institutional relationship that could be perceived as a potential conflict of interest.

Funding Information

This research was conducted without any external financial support from public, commercial, or non-profit funding agencies. All computational resources and experimental work were carried out independently by the authors as part of their research activities, without external financial support.

REFERENCES

- [1] K. Rizman Žalik and M. Žalik, "A review of federated learning in agriculture," *Sensors*, vol. 23, no. 23, p. 9566, 2023.
- [2] D. Bacciu, S. Chessa, C. Gallicchio, and A. Micheli, "On the need of machine learning as a service for the Internet of Things," in *Proc. 1st Int. Conf. Internet of Things and Machine Learning*, 2017.
- [3] H. Assem, L. Xu, T. S. Buda, and D. O'Sullivan, "Machine learning as a service for enabling Internet of Things and people," *Personal and Ubiquitous Computing*, 2016.
- [4] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečný, S. Mazzocchi, H. B. McMahan, T. Van Overveldt, D. Petrou, D. Ramage, and J. Roselander, "Towards federated learning at scale: System design," in *Proc. 2nd SysML Conf.*, 2019. "latex
- [5] L. Ge, H. Li, X. Wang, and Z. Wang, "A review of secure federated learning: Privacy leakage threats, protection technologies, challenges and future directions," *Neurocomputing*, vol. 561, Art. no. 126897, 2023, doi: 10.1016/j.neucom.2023.126897. "
- [6] L. García, L. Parra, J. M. Jiménez, J. Lloret, and P. Lorenz, "IoT-based smart irrigation systems: An overview on the recent trends on sensors and IoT systems for irrigation in precision agriculture," *Sensors*, vol. 20, no. 4, p. 1042, 2020.
- [7] Soil Science Division Staff, *Soil Survey Manual*, C. D. Ditzler, K. Scheffe, and H. C. Monger, Eds. Washington, D.C.: U.S. Department of Agriculture, USDA Handbook 18, 2017.

- [8] Soil Science Division Staff, *Soil Survey Manual*. Washington, D.C.: U.S. Department of Agriculture, 1993, pp. 63–65.
- [9] S. Sengupta, P. Basak, and S. Saikat, “Distributed machine learning in IoT: The future of distributed edge computing,” *IEEE Trans. Industrial Informatics*, 2020.
- [10] P. Kairouz, H. B. McMahan, and others, “Advances and open problems in federated learning,” *Foundations and Trends in Machine Learning*, vol. 11, nos. 3–4, pp. 185–383, 2019.
- [11] H. B. McMahan, E. Moore, D. Ramage, and B. A. Y. Arcas, “Communication-efficient learning of deep networks from decentralized data,” in *Proc. 20th Int. Conf. Artificial Intelligence and Statistics*, 2017.
- [12] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, “Federated learning: Challenges, methods, and future directions,” *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.
- [13] V. Smith, C. K. Chiang, M. Sanjabi, and A. Talwalkar, “Federated multi-task learning,” in *Proc. 31st Int. Conf. Neural Information Processing Systems*, 2017.
- [14] J. Park and K. M. Sim, “Blockchain-based dynamic federated learning for secure smart agriculture,” *IEEE Access*, vol. 8, pp. 139109–139118, 2020.
- [15] J. Konečný, H. B. McMahan, F. X. Yu, and others, “Federated learning: Strategies for improving communication efficiency,” in *Proc. 19th Int. Conf. Neural Information Processing Systems*, 2016.
- [16] “Ebergötzen soil dataset,” Kaggle, dataset of soil composition measurements from the Ebergötzen region, Germany. ““latex
- [17] Marwen Ghabi, “Ebergötzen soil texture dataset – USDA soil texture classification,” GitHub, 2025. Available: <https://github.com/Marwenghabi/ebertgotzen-soil-texture-usda12>.