SoftwareX 23 (2023) 101505

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

OpenFL-XAI: Federated learning of explainable artificial intelligence models in Python

Mattia Daole [a,*], Alessio Schiavo [a,b], José Luis Corcuera Bárcena [a], Pietro Ducange [a], Francesco Marcelloni [a], Alessandro Renda [a]

[a] Department of Information Engineering, University of Pisa, Largo Lucio Lazzarino 1, 56122 Pisa, Italy
[b] LogObject AG, Ambassador House Thurgauerstrasse 101 A, CH-8152 Opfikon, Switzerland

ARTICLE INFO

Article history:
Received 16 June 2023
Received in revised form 29 July 2023
Accepted 15 August 2023

Dataset link: https://github.com/Unipisa/OpenFL-XAI

Keywords:
Federated learning
Explainable AI
Rule-based systems
Linguistic fuzzy models

ABSTRACT

Artificial Intelligence (AI) systems play a significant role in manifold decision-making processes in our daily lives, making trustworthiness of AI more and more crucial for its widespread acceptance. Among others, privacy and explainability are considered key requirements for enabling trust in AI. Building on these needs, we propose a software for Federated Learning (FL) of Rule-Based Systems (RBSs): on one hand FL prioritizes user data privacy during collaborative model training. On the other hand, RBSs are deemed as interpretable-by-design models and ensure high transparency in the decision-making process. The proposed software, developed as an extension to the Intel® OpenFL open-source framework, offers a viable solution for developing AI applications balancing accuracy, privacy, and interpretability.

Code metadata

Current code version	v1.0.0
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-23-00391
Code Ocean compute capsule	Not Applicable
Legal Code License	Apache license 2.0
Code versioning system used	Git
Software code languages, tools, and services used	Python
Compilation requirements, operating environments & dependencies	Docker, Python 3
If available Link to developer documentation/manual	Not Applicable
Support email for questions	mattia.daole@phd.unipi.it

1. Motivation and significance

AI solutions are being widely exploited in several high stakes domains such as medicine, law and smart mobility. Decisions derived from AI systems may ultimately affect humans' lives. As a consequence, there is an emerging and urgent need for understanding how such decisions are taken, as also witnessed by the notable momentum achieved by eXplainable AI (XAI) in recent years [1].

Besides explainability, another concern, somewhat hindering the even more massive adoption of AI systems, pertains privacy of the huge amount of data which AI heavily relies on. Indeed, in scenarios where preserving privacy of raw data is a crucial requirement, traditional Machine Learning (ML) approaches are no longer viable, since they require to access data samples for training AI models. Thus, alternative paradigms have to be investigated. In this regard, Federated Learning (FL) [2] allows multiple parties to collaboratively train a global model while ensuring privacy-by-default, as raw data are neither disclosed nor transferred.

While FL is today extensively studied and various frameworks for FL exist [3], most solutions revolve around the original proposal of Federated Averaging (FedAvg) as a protocol for executing Stochastic Gradient Descent (SGD) optimization of (Deep) Neural Networks (NN) in a round-based federated procedure. However, NNs are generally considered opaque models due to their huge number of parameters and non-linear modeling. Thus, providing

* Corresponding author.
E-mail address: mattia.daole@phd.unipi.it (Mattia Daole).

https://doi.org/10.1016/j.softx.2023.101505

explanations for the outputs of NNs is still considered an open challenge.

In the spirit of ensuring a greater explainability of models than NNs, some recent works have investigated FL of classical ML models. In [4] the authors have proposed a federated version of AdaBoost, labeled as AdaBoost.F, to generate an ensemble of traditional ML models, such as decision trees. Furthermore, the work in [5] argues on the integration of AdaBoost.F into Intel® Open Federated Learning (OpenFL) framework [6,7]. Finally, a federated Random Survival Forest has been proposed in [8]. The aforementioned works deal with the FL of ensembles of ML models. These models are certainly more interpretable than opaque models like NNs. Nevertheless, they cannot be considered as inherently interpretable. The limited number of works exploring the adoption of FL of inherently interpretable models results in a shortage of ready-to-use FL frameworks that fulfill, at the same time, both privacy and explainability requirements.

In the field of XAI, Fuzzy RBSs (FRBSs) [9,10] are considered among the most interpretable models: they consist in collections of rules, where each rule is expressed in the format "IF *antecedent* THEN *consequent*" and uses linguistic terms whose meaning is expressed by fuzzy sets defined on the input–output variable domains. The antecedent typically combines one or more fuzzy sets using logical operators and identifies the space region to which the rule applies. The format of the consequent varies depending on the ML task at hand: in regression tasks, the consequent typically holds a scalar value (zero-order) or a linear combination of the input variables (first-order). Although research efforts at the interface between FL and XAI are proliferating [11–17], to the best of our knowledge there is currently no framework supporting FL of FRBSs. A framework developed by IBM [18] supports FL of interpretable models (i.e., decision trees), but it is not open-source.

In this paper, we propose OpenFL-XAI, an extension to the aforementioned open-source Intel® OpenFL framework,

providing user-friendly support for learning FRBSs as explainable-by-design models in a federated fashion. OpenFL-XAI allows users to easily: (i) add the code to train their own XAI model (FRBS) in a federated fashion; (ii) provide a new custom logic to aggregate the rules generated by the single participants to the federation, and to generate the federated model of aggregated rules; (iii) execute FL process of XAI model with a custom number of participants; (iv) store local and federated XAI models.

Notably, this extension comes as an addition on top of the native features provided by OpenFL and supports FL of any model that can be represented as a collection of *if-then* rules.

Users can exploit OpenFL-XAI to easily compare, in the federated setting, opaque models (e.g., NNs, natively supported in OpenFL) and explainable-by-design models, both in terms of accuracy and interpretability.

The rest of this paper is structured as follows. In Section 2, we provide an overview of OpenFL and a thorough description of our extension, OpenFL-XAI, focusing on its components and functionalities.

In Section 3, a comprehensive use case example of FL of FRBSs for a regression task is presented. Section 4 discusses the benefits entailed by the proposed solution. Finally, in Section 5 we draw some conclusions.

2. Software description

OpenFL is an open-source framework developed by Intel® Labs within a research project with the University of Pennsylvania. It is mostly written in Python and distributed via *pip*, *conda*, and *Docker* packages. OpenFL enables multiple participants (i.e., data owners), possibly dislocated across different sites, to collaboratively train an ML model exploiting their own hardware infrastructure and preserving the privacy of their raw data. Indeed, only the parameters of the locally trained models are exchanged through secure remote procedure calls (RPCs), possibly enabling mutually-authenticated transport layer security (TLS) connections. Although it is described as "ML framework agnostic" (wrapping PyTorch,[1] Tensorflow[2] and Keras[3] libraries), it is natively oriented towards deep learning models: nevertheless it is particularly suited to our purpose as it is highly modular and can be extended with support to any kind of ML model while still relying on the existing back-end for the FL process.

Fig. 1. OpenFL main components overview. The Collaborator sends remote procedure calls to the Collaborator through the TCP Client to retrieve the list of tasks. Similarly, the Aggregator receives model and metric updates for aggregation through the TCP Server. More details on the communication can be found in [6]. Components extended to support FL of XAI models are highlighted with yellow borders.

In the following we first introduce a high-level overview of the OpenFL software architecture with the essential details of its main software modules. Then, we describe the OpenFL-XAI software architecture by explaining which components were modified to fulfill our goal of FL of XAI models and which extended software functionalities were introduced.

2.1. OpenFL software architecture

Fig. 1 presents a high-level overview of the OpenFL software architecture. OpenFL has two types of entities, namely Collaborator and Aggregator. For each entity of the framework, the main components are depicted. In an FL process multiple Collaborator nodes (i.e., data owners) directly connect to a central Aggregator node (orchestrating server), according to a star, or centralized, FL communication topology.

Before the FL process begins, all participant nodes must be provided with a collection of configuration files named *workspace*. The workspace comprises the following items:

[1] https://pytorch.org/, visited June 2023

[2] https://www.tensorflow.org/, visited June 2023

[3] https://keras.io/, visited June 2023

- the code base for implementing the functionalities of the specific ML model (i.e., initialization, data loading, training, validation, and testing);
- the FL Plan: a YAML[4] file with the FL process configuration including aggregator IP address, FL process parameters, ML model parameters;
- a folder containing the public certificate and the private key (each node has its pair) needed to establish the TLS connections.

Moreover, the Aggregator and Collaborator nodes have a YAML file with, respectively, the list of the participant nodes and the path to the private data location.

As a first step, the Aggregator node is started and its components are initialized by parsing the FL Plan. The Aggregator is the central entity of the federation whose main tasks consist in coordinating the Collaborators, dispatching the FL tasks, and aggregating Collaborators' local models into the federated one, according to a certain aggregation function. Once instantiated, the Aggregator waits for the Collaborator nodes to invoke RPCs to retrieve the list of tasks to be executed. Examples of tasks include model training and validation.

When the Collaborator nodes are started, their components are initialized parsing the FL Plan. The Collaborator is, by design, the only entity having access to the local data. Until the End Of Federation (EOF) is reached every Collaborator retrieves the list of assigned tasks from the Aggregator throughout each federation round. For each task, the Collaborator loads local data with the Data Loader component. Subsequently, the task is performed through the Task Runner component, which contains the task implementation. The results, including metrics and local model parameters, are then transmitted to the Aggregator. The Collaborator nodes conclude their operations upon receiving an EOF signal from the Aggregator.

Whenever the Aggregator receives updates from a Collaborator, it verifies if all Collaborators have finalized their tasks, i.e., whether the round can be considered completed. If yes, the Aggregator combines local contributions through a dedicated policy (e.g., FedAvg) and evaluates whether the FL process can terminate. If yes, an EOF signal is sent to the Collaborators.

2.2. OpenFL-XAI software architecture

In Fig. 1, we highlighted with yellow borders the OpenFL components which were extended in OpenFL-XAI. A more detailed view of this extension is presented in the UML diagram shown in Fig. 2.

The objective of the extension is to enable FL of rule-based models that adhere to the format introduced in Section 1. For space limits, in this paper we do not elaborate on the theoretical aspects of FL of XAI models (the interested reader may refer to the relevant literature [11,12]). We modified the OpenFL framework taking two non-functional requirements into account. First, we aimed to make only essential modifications to the original OpenFL framework for enabling FL of XAI models, while preserving OpenFL core functionalities and data structures. Second, our aim was to provide users with a user-friendly, readily applicable solution for training their own (F)RBSs in a federated manner. As OpenFL is mainly oriented towards NNs, its FL workflow (and communication layer, accordingly) revolves around the exchange of tensors for sharing NN weights. Our primary modifications are therefore centered around leveraging these tensors to exchange rules rather than NN weights.

[4] https://yaml.org/, visited June 2023

With no loss of generality, we hereby consider first-order Takagi Sugeno Kang (TSK) FRBS [19]. The generic kth rule ($k = 1, \ldots, K$) is expressed in the following form:

$$R_k : \textbf{IF} \quad X_1 \text{ is } A_{1,j_{k,1}} \quad \textbf{AND} \quad \ldots \quad \textbf{AND} \quad X_F \text{ is } A_{F,j_{k,F}} \quad \textbf{THEN} \quad y_k = \gamma_{k,0} + \sum_{f=1}^{F} \gamma_{k,f} \cdot x_f \tag{1}$$

where F is the total number of input variables, $A_{f,j_{k,i}}$ is the jth fuzzy set of the fuzzy partition over the fth input variable X_f, and $\gamma_{k,f}$ (with $f = 0, \ldots, F$) are the coefficients of the linear model, which is used to evaluate the associated output y_k.

The rule antecedents are codified as a $K \times F$ matrix of integer numbers where the element (k, f) represents the index of the fuzzy set for the fth variable in the kth rule. Similarly, the consequents of the rules are codified as a $K \times (F + 1)$ matrix of real numbers where the element (k, f) represents the value of the coefficient for the fth variable of the kth rule (the additional column hosts the $\gamma_{k,0}$ values). Lastly, each rule has an associated rule weight: such weights are represented as a vector of K real numbers, where the element w_k is the weight of the kth rule.

The modifications to OpenFL that characterize OpenFL-XAI are listed in the following:

- The `AggregatorXAI` class extends the `Aggregator` base class by incorporating the necessary logic to parse the tensors carrying the rules extracted from the Collaborators' local models.
- The `AggregationFunctionXAI` extends the abstract class `AggregationFunction` to implement the aggregation function for the interpretable-by-design rule-based models. A user can easily leverage its own FL aggregation policy by implementing the aggregation function and properly referring to it in the FL Plan.
- Similarly, on the Collaborator side, the `CollaboratorXAI` class extends the `Collaborator` base class. This extension ensures proper transmission of the local RBS through tensors.
- The `TensorDBXAI` class extends the `TensorDB` base class. `TensorDB` is the implementation of an in-memory database of tensors. Each Collaborator and Aggregator has its own `TensorDB`. `TensorDBXAI` provides the logic to properly exchange and parse the tensors as rule containers.
- The `XAIDataLoader` extends the `DataLoader` base class: this was needed to load training, testing, and validation data in the proper format and data structures as needed by the specific ML framework.
- The `TaskRunnerXAI` class encompasses the implementations of all tasks associated with a particular ML model: each model needs its own class.
- The `Model` abstract base class defines a set of mandatory methods, which must be implemented in any custom RBS class. In Fig. 2 the class `TSK` (associated with the dedicated "FL Plan TSK") refers to the specific model adopted in our illustrative example and thoroughly discussed in Section 3. Users aiming to train their own RBS in a federated manner need to provide an extension to the `Model` abstract class.

2.3. Software functionalities

OpenFL-XAI serves as an extension to OpenFL, inheriting its primary functionalities, which have been previously illustrated. However, OpenFL-XAI expands upon these functionalities by incorporating support for RBSs that are not directly accommodated in the original version. As previously mentioned, this work specifically focuses on FRBSs, which can be considered as a generalization of traditional RBSs.

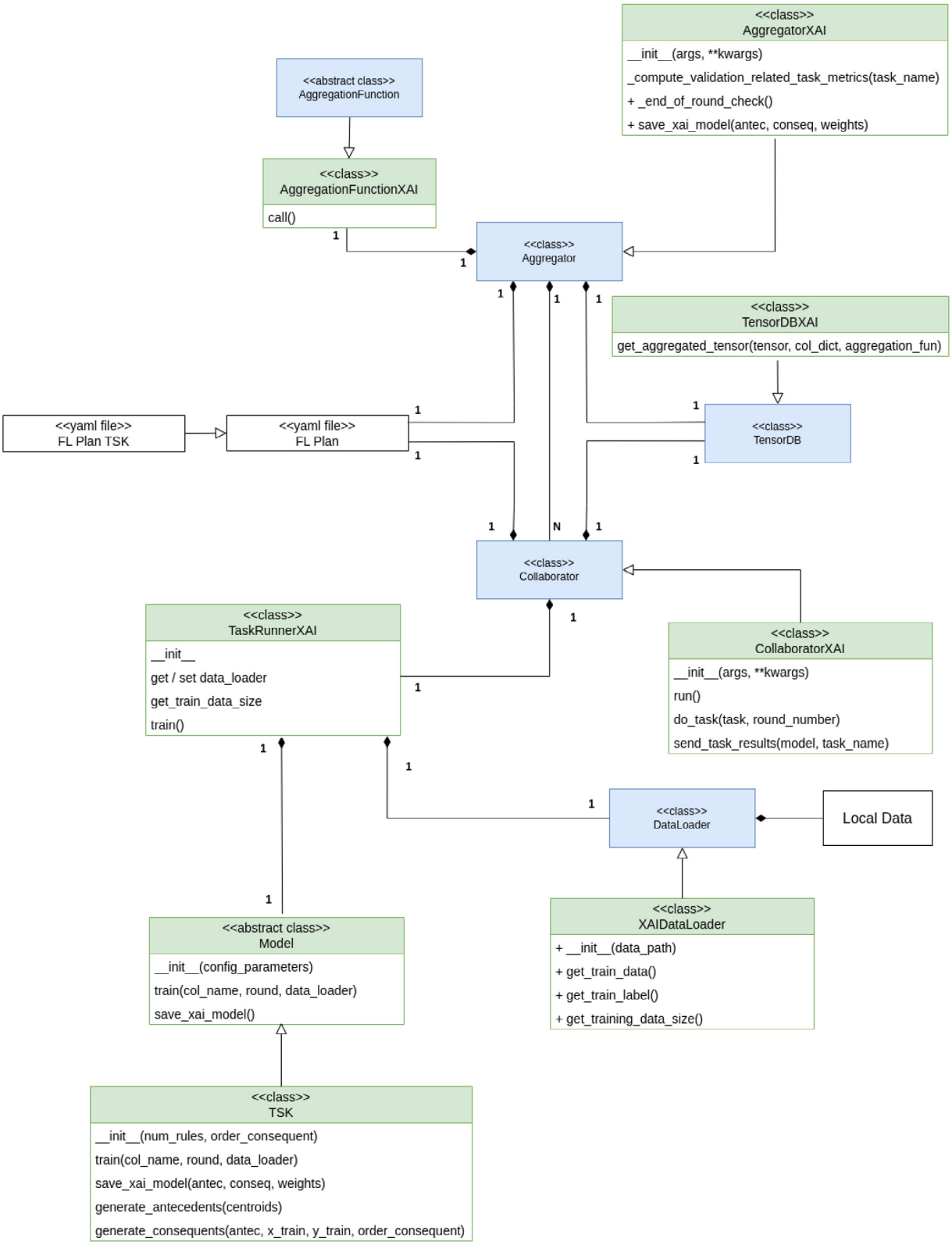

Fig. 2. UML Class Diagram of OpenFL-XAI. Classes from original OpenFL are represented in blue, whereas those extended or added in OpenFL-XAI are represented in green.

3. An illustrative example

In this example, we consider the use case of FL of a TSK-FRBS on a public regression dataset, namely *Mortgage* [20], purposely partitioned over five Collaborator nodes. For FL of TSK-FRBSs we employ the methodology proposed in [11]. In a nutshell, the idea behind the aggregation logic is the following. The Aggregator combines, by juxtaposition, the local rule sets collected from Collaborators. Furthermore, since these rule sets reflect knowledge extracted from different local raw data, some conflicts may arise

```
xai_tsk_frbs/
├── cert
│   ├── client
│   └── server
├── data
├── plan
│   ├── cols.yaml
│   ├── data.yaml
│   └── plan.yaml
├── requirements.txt
└── src
    ├── Fed_XAI_Classes
    │   ├── aggregation_function_xai.py
    │   ├── tensor_db.py
    │   ├── xai_aggregator.py
    │   ├── xai_collaborator.py
    │   ├── xai_data_loader.py
    │   └── xai_runner.py
    └── model
        ├── fuzzySystem.py
        ├── model.py
        └── tsk.py
```

Fig. 3. OpenFL-XAI workspace for TSK-FRBS.

and must be adequately handled. Conflicting rules have identical antecedents, but different consequents. The Aggregator solves the conflict by creating a single rule with the common antecedent, and as consequent the weighted average of the consequent coefficients of the conflicting rules, weighted by the corresponding rule weights.

Fig. 3 shows the specific structure of the workspace employed in this example.

As outlined in Section 2.1, the federation workspace is shared among all participating nodes in the federation. The folder *src/Fed_XAI_Classes* contains the set of customized framework components enabling FL of RBS. Additionally, the */src/model* directory hosts the source code specifically associated with the TSK-FRBS. It is worth highlighting that the class implementing the custom ML model extends the abstract class `Model` (located in */src/model/model.py*).

Fig. 4 illustrates the portion of the FL `Plan` (*/plan/plan.yaml*) with the configuration for the `TaskRunnerXAI` component. The `template` property stores the file path of the customized `TaskRunnerXAI` class. The `dict_model` property holds a dictionary indicating both the path and the class name of the specific ML model. Finally, the `dict_parameters` property hosts a dictionary containing the values of the ML model hyperparameters (number of rules and order of the consequent).

Each participant must provide a valid certificate and key pair in the workspace to establish a mutually authenticated communication channel. The certificate and key pair of the Aggregator and each Collaborator must be named as `agg_<aggregatorname>` and `col_<collaboratorname>` and stored in the */cert/server* and */cert/client* folders, respectively. Additionally, each participant must have the public certificate of the Certification Authority (CA) to validate the other parties' identities. This file must be named `cert_chain.crt` and must be placed under the */cert* folder. In our example, we adopt a private *OpenSSL* CA and self-signed certificates. Finally, the Aggregator needs to be aware of the Collaborators' names, written in the file */plan/cols.yaml*.

Within the workspace of each Collaborator, the *data* directory stores the local training and test data, so that they can be properly loaded through the `XAIDataLoader` class in */src/Fed_XAI_Classes/xai_data_loader.py* module.

Once each participant node has the workspace properly configured, the FL process can be initiated by issuing the following commands:

- On the terminal of the Aggregator node, navigate to the federation workspace folder and execute the command

$$\$\ \texttt{fx aggregator start} \tag{2}$$

  This command will start an Aggregator instance that waits for the Collaborators to establish connections.
- On each Collaborator node, navigate to the workspace folder and execute the command

$$\$\ \texttt{fx collaborator start -n <collaborator\_name>} \tag{3}$$

When the FL process is over, the Aggregator saves the federated model in its file system in three binary files in NumPy format (.npy) before terminating its execution. These three files contain, respectively, the antecedents, consequents and rule weights of the generated TSK-FRBS. Details about the results of the FL process, in terms of model structure and its usage for performing inference on local data, are thoroughly described in the `IllustrativeExample.md` file of the GitHub repository.

In our experiment, each participant is deployed inside a *Docker* container to ease reproducibility and empower portability. Lastly, a minimal command line interface has been implemented, which facilitates users in replicating the experiment. By adhering to the instructions displayed on the screen, users can effortlessly execute the FL process and obtain the trained model.

4. Impact

The preservation of privacy and the transparency of the models are considered as imperative requisites for establishing trust in AI systems. Consequently, the integration of FL and XAI techniques serves as a significant facilitator in numerous safety-critical domains reliant on AI applications. One illustrative application domain is healthcare, as shown in [21], where FL and XAI techniques are used in an electrocardiogram monitoring use case, thus guaranteeing the privacy of users' data while ensuring the interpretability of the system. Similarly, in [22], the authors leverage XAI techniques within a Vehicle-to-Everything (V2X) use case to assess the contributions made by each participant in an FL process for trajectory and motion prediction. In the research domain, OpenFL-XAI provides a convenient tool for performing comprehensive experimental analyses aimed at comparing opaque models (for instance DNNs, natively supported in OpenFL) against explainable-by-design AI models within a federated environment. In the software domain, OpenFL-XAI can be considered a user-friendly framework supporting a rapid development of XAI models by exploiting FL.

Currently, OpenFL-XAI is actively employed within Hexa-X, the European Union's flagship 6G project.[5] Its deployment in this project supports research, development, and demonstration activities concerning the FL of XAI models, which has been recently awarded as key innovation by the EU Innovation Radar.[6]

5. Conclusions

In this paper, we have presented OpenFL-XAI, an extension to the Intel® OpenFL framework that facilitates the federated learning (FL) of inherently interpretable AI models. The architecture and functionalities of OpenFL-XAI have been thoroughly described by highlighting the major enhancements over the original software. Furthermore, an illustrative example of FL of an XAI model, namely a first-order TSK-FRBS, has been discussed,

[5] https://hexa-x.eu/, visited June 2023

[6] https://www.innoradar.eu/innovation/45988, visited June 2023

```
task_runner:
  settings:
    dict_parameters: {"num_rules":30, "order_consequent":"first"}
    dict_model: {"model_class_name":"TSK", "model_module_path":"src.model.tsk"}
  template: src.Fed_XAI_Classes.xai_runner.XAITaskRunner
```

Fig. 4. TaskRunner details in the *plan.yaml* configuration file.

emphasizing crucial configuration details. Envisioning a growing interest in the domains of FL and XAI techniques in the coming years, the software implementation has been released under an open-source license: this choice encourages and enables further development and innovation within the field, with the overarching goal of promoting the adoption of privacy-preserving and transparent AI tools.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Francesco Marcelloni, Pietro Ducange reports financial support was provided by Hexa-X (Grant Agreement no. 10101595). Francesco Marcelloni, Pietro Ducange reports financial support was provided by PNRR - FAIR (PE0000001). Alessandro Renda reports financial support was provided by PNRR - THE (CUP I53C2200078000). Mattia Daole, Alessio Schiavo, Jose Luis Corcuera Barcena, Pietro Ducange, Francesco Marcelloni, Alessandro Renda reports was provided by FoReLab and CrossLab projects (Department of Excellence).

Data availability

Code is available at https://github.com/Unipisa/OpenFL-XAI.

Acknowledgments

This work has been partly funded by: i) the European Commission through the project Hexa-X (Grant Agreement no. 101015956) under the H2020 programme, ii) the PNRR - M4C2 - Investimento 1.3, Partenariato Esteso PE00000013 - "FAIR - Future Artificial Intelligence Research" - Spoke 1 "Human-centered AI" and iii) the PNRR "Tuscany Health Ecosystem" (THE) (Ecosistemi dell'Innovazione) - Spoke 6 - Precision Medicine & Personalized Healthcare (CUP I53C22000780001) under the NextGeneration EU programme, and by the Italian Ministry of University and Research (MUR), Italy in the framework of the FoReLab and CrossLab projects (Departments of Excellence). Authors would like to thank Intel® and, in particular, Ing. Dario Sabella for providing hardware support and for the fruitful discussions.

References

[1] Barredo Arrieta A, Díaz-Rodríguez N, Del Ser J, Bennetot A, Tabik S, Barbado A, et al. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. Inf Fusion 2020;58:82–115.

[2] Yang Q, Liu Y, Chen T, Tong Y. Federated machine learning: Concept and applications. ACM Trans Intell Syst Technol 2019;10(2):1–19.

[3] Kholod I, Yanaki E, Fomichev D, Shalugin E, Novikova E, Filippov E, et al. Open-source federated learning frameworks for IoT: A comparative review and analysis. Sensors 2021;21(1). http://dx.doi.org/10.3390/s21010167, URL https://www.mdpi.com/1424-8220/21/1/167.

[4] Polato M, Esposito R, Aldinucci M. Boosting the federation: Cross-silo federated learning without gradient descent. In: 2022 international joint conference on neural networks. 2022, p. 1–10. http://dx.doi.org/10.1109/IJCNN55064.2022.9892284.

[5] Mittone G, Riviera W, Colonnelli I, Birke R, Aldinucci M. Model-agnostic federated learning. 2023, arXiv:2303.04906.

[6] Reina GA, et al. OpenFL: An open-source framework for Federated Learning. 2021, http://dx.doi.org/10.48550/arXiv.2105.06413, CoRR abs/2105.06413 URL https://arxiv.org/abs/2105.06413.

[7] Foley P, Sheller MJ, Edwards B, Pati S, Riviera W, Sharma M, et al. OpenFL: the open federated learning library. Phys Med Biol 2022. http://dx.doi.org/10.1088/1361-6560/ac97d9, URL http://iopscience.iop.org/article/10.1088/1361-6560/ac97d9.

[8] Archetti A, Matteucci M. Federated survival forests. 2023, arXiv:2302.02807.

[9] Magdalena L. Fuzzy rule-based systems. In: Kacprzyk J, Pedrycz W, editors. Springer handbook of computational intelligence. Berlin, Heidelberg: Springer Berlin Heidelberg; 2015, p. 203–18.

[10] Fernandez A, Herrera F, Cordon O, Jose del Jesus M, Marcelloni F. Evolutionary fuzzy systems for explainable artificial intelligence: Why, when, what for, and where to? IEEE Comput Intell Mag 2019;14(1):69–81. http://dx.doi.org/10.1109/MCI.2018.2881645.

[11] Corcuera Bárcena JL, Ducange P, Ercolani A, Marcelloni F, Renda A. An approach to federated learning of explainable fuzzy regression models. In: 2022 IEEE international conference on fuzzy systems. IEEE; 2022, p. 1–8. http://dx.doi.org/10.1109/FUZZ-IEEE55066.2022.9882881.

[12] Bárcena JLC, Daole M, Ducange P, Marcelloni F, Renda A, Ruffini F, et al. Fed-XAI: Federated learning of explainable artificial intelligence models. In: XAI.It 2022: 3rd Italian workshop on explainable artificial intelligence, co-located with AI*IA 2022. 2022, URL https://ceur-ws.org/Vol-3277/paper8.pdf.

[13] Bechini A, Daole M, Ducange P, Marcelloni F, Renda A. An application for federated learning of XAI models in edge computing environments. In: 2023 IEEE international conference on fuzzy systems. 2023 [in press].

[14] Bárcena JLC, Ducange P, Marcelloni F, Renda A, Ruffini F, Schiavo A. Federated TSK models for predicting quality of experience in B5G/6G networks. In: 2023 IEEE international conference on fuzzy systems. 2023 [in press].

[15] Renda A, Ducange P, Marcelloni F, Sabella D, Filippou MC, Nardini G, et al. Federated learning of explainable AI models in 6G systems: Towards secure and automated vehicle networking. Information 2022;13(8):395.

[16] Zhu X, Wang D, Pedrycz W, Li Z. Horizontal federated learning of Takagi-Sugeno fuzzy rule-based models. IEEE T Fuzzy Syst 2021.

[17] Wilbik A, Grefen P. Towards a federated fuzzy learning system. In: 2021 IEEE int'l conf on fuzzy systems. 2021, p. 1–6.

[18] Ludwig H, Baracaldo N, Thomas G, Zhou Y, Anwar A, Rajamoni S, et al. IBM federated learning: an enterprise framework white paper V0.1. 2020, arXiv:2007.10987.

[19] Takagi T, Sugeno M. Fuzzy identification of systems and its applications to modeling and control. IEEE T Syst Man Cyb 1985;(1):116–32.

[20] Alcala-Fdez J, Fernández A, Luengo J, Derrac J, Garc'ia S, Sanchez L, et al. KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. J Mult-Valued Logic Soft Comput 2010;17:255–87.

[21] Saraswat D, Bhattacharya P, Verma A, Prasad VK, Tanwar S, Sharma G, et al. Explainable AI for healthcare 5.0: Opportunities and challenges. IEEE Access 2022;10:84486–517. http://dx.doi.org/10.1109/ACCESS.2022.3197671.

[22] Rjoub G, Bentahar J, Wahab OA. Explainable AI-based federated deep reinforcement learning for trusted autonomous driving. In: 2022 international wireless communications and mobile computing. 2022, p. 318–23. http://dx.doi.org/10.1109/IWCMC55113.2022.9824617.