Research and Innovation Action
HORIZON-CL6-2021-FARM2FORK-01
TOPIC: HORIZON-CL6-2021-FARM2FORK-01-18

NESTLER
oNe hEalth SusTainabiLity partnership between
EU-AFRICA for food sEcuRity

Deliverable D4.1
Initial NESTLER backend implementation of AI algorithms

Authors	A. Nchange Kouotou, Th. Zahariadis, S. Bourou, A. Skias, G. Athanasiou, G. Pantelide, A. Baglatzi, S. Charamoutsou, K. Pramataris, P. Karkazis, A. Tomaras
Nature	Report
Dissemination	PU
Version	2.0
Status	Final
Delivery Date (DoA)	M15
Actual Delivery Date	14/082024

Keywords	Federated Machine Learning, Artificial intelligence, Machine Learning, Risk assessment, Impact assessment, Drought/ Flood, Prediction, Weather, Climate data, Satellite data, GIS, crop yield, livestock, insect, NESTLER Architecture.
Abstract	Deliverable D4.1 outlines the initial design and implementation of the NESTLER AI platform as well as a description of Federated Learning approach that is considered to be used in NESTLER. It also details some initial applications of AI algorithms that will be tested and integrated on the NESTLER platform. Moreover, the NESTLER Integration Framework and Tools are defined and described, each of which is able to cater to all of NESTLER's subcomponents. Finally, it presents the NESTLER reference architecture.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No 957246

DISCLAIMER

	Participant organisation name	*Short*	*Country*
01	SYNELIXIS SOLUTIONS S.A.	SYN	EL
02	CloudEO AG	CEO	DE
03	RINISOFT LTD	RINIS	BU
04	EBOS TECHNOLOGIES LIMITED	eBOS	CY
05	STICHTING IDH SUSTAINABLE TRADE INITIATIVE	IDH	NL
06	ZANASI ALESSANDRO SRL	Z&P	IT
07	AGRIX TECH SARL	AGRI	CM
08	CONSERVATION THROUGH PUBLIC HEALTH	CTPH	UG
09	THE INTERNATIONAL CENTRE OF INSECT PHYSIOLOGY AND ECOLOGY LBG	ICIPE	KE
10	ETHIOPIAN INSTITUTE OF AGRICULTURAL RESEARCH	EIAR	ET
11	RWANDA AGRICULTURE AND ANIMAL RESOURCES DEVELOPMENT BOARD	RAB	RW
12	INTERNATIONAL INSTITUTE OF TROPICAL AGRICULTURE	IITA	NG
13	MANA BIOSYSTEMS LIMITED	MANA	UK
14	UNIVERSITY COLLEGE LONDON	UCL	UK
15	RINISOFT LTD	RINIS	BG

ACKNOWLEDGEMENT

This document is a deliverable of NESTLER project. This project has received funding from the European Union's Horizon Research and innovation programme under grant agreement Nº 101060762.
Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Executive Agency, while neither the European Union nor the granting authority can be held responsible for any use that may be made of the information it contains.

Document History

Version	Date	Contributor(s)	Description
v0.1	23/10/2023	AGRI	ToC/Initial Version
v0.2	17/11/2023	AGRI, EBOS, CEO	Initial Input about AI backend algorithms
v0.3	24/11/2023	SYN	Input about FL and reference architecture
v0.4	15/12/2023	SYN, AGRI, EBOS, CEO	Updated input
v0.5	20/12/2023	CEO, EBOS	Peer-review
V0.6	21/12/2023	AGRI	Minor changes to entire draft
v0.7	22/12/2023	SYN	Minor changes to entire draft
v1.0	22/12/2023	SYN	Final version after quality checks
V2.0	14/08/2024	AGRI	Updated after Review Comments. Added section 5.2 for Smart irrigation section

Document Reviewers

Date	Reviewer's name	Affiliation
20/12/2023	Georgia Pantelide	EBOS
20/12/2023	Alkyoni Baglatzi	CEO

Table of Contents

1. List of Figures

2. List of Tables

3. Definitions, Acronyms and Abbreviations

AI	Artificial Intelligence
AP	Average Precision
API	Application Programming Interface
CA	Certificate Authority
CD	Continuous Deployment
CI	Continuous Integration
CIFAR	Canadian Institute For Advanced Research
CNN	Convolutional Neural Network
COCO	Common Objects in Contex
CSI	Cubic Spline Interpolation
DAG	Directed Acyclic Graph
DBMS	DataBase Management System
DevOps	Development, and Operations
DevSecOps	Development, Security, and Operations
DL	Deep Learning
DoS	Denial of Service
DP	Differential Privacy
DP-SGD	Differentially Private Stochastic Gradient Descent
DSL	Domain-Specific Language
EVI	Enhanced Vegetation Index
FAO	Food and Agriculture Organization
FATE	Federated AI Technology Enabler
FL	Federated Learning
FLARE	Federated Learning Application Runtime Environment
FML	Federated Machine Learning
FPS	Frames Per Second
FQDN	Fully Qualified Domain Name
GAN	Generative Adversarial Network
GD	Gradient Descent
GDP	Global Differential Privacy
GIS	Geographic Information System
GPU	Graphics Processing Unit
HE	Homomorphic Encryption
IDS	Intrusion Detection System

IoT	Internet of Things
IoU	Intersection over Union
IPD	Individual Participant Data
JSON	JavaScript Object Notation
LDP	Local Differential Privacy
LSA	Least Square Approximation
MA	Moving Average
MAP	Mean Average Precision
MIT	Massachusetts Institute of Technology
ML	Machine Learning
MPC	Multi-Party Computation
MSE	Mean Squared Error
NAIF	NESTLER AI framework
NDVI	Normalized Difference Vegetation Index
NDWI	Normalized Difference Water Index
PATE	Private Aggregation of Teacher Ensembles
PKI	Public Key Information
POC	Proof of Concept
QA	Quality Assurance
R2L	Remote-to-Local
RMSE	Root Mean Square Error
RPC	Remote Procedure Call
RSA	Rivest–Shamir–Adleman
SAST	Static Application Security Testing
SCM	Source Code Management
SEO	Search Engine Optimization
SGD	Stochastic Gradient Descent
SMC	Secure Multiparty Computation
SNNPR	Southern Nations, Nationalities, and Peoples' Region
SPEI	Standardized Precipitation-Evapotranspiration Index
SSL	Secure Sockets Layer
TFF	TensorFlow Federated
TLS	Transport Layer Security
U2R	User-to-Root
UC	Use Case
UI	User interface
VI	Vegetation Index
VOC	Visual Object Class

WP	Work Package
YAML	Yet Another Markup Language
YOLO	You Only Look Once
ZKP	Zero-Knowledge Proofs

4. Executive Summary

The NESTLER project team has chosen *Federated Machine Learning (FML)* technique as the solution for the NESTLER backend implementation of AI algorithms. FML is a technique that leverages a group of nodes to collectively train a machine learning model without revealing their individual data to each other. This technique allows the final model to be trained on a wider range of data sets in the current context where most organizations are reluctant to share their data.

To set up the NESTLER AI platform, NESTLER has developed NAIF which is an FML framework that combines the functionalities of the best frameworks in their respective fields. Thanks to an appropriate choice and combination of frameworks, the NESTLER AI platform is both reliable and flexible.

Additionally, a structured yet adaptable integration framework for NESTLER is defined, introducing the DevSecOps approach—encompassing development, security, and operations—and outlining the initial framework components such as Source Code Management, CI/CD, Issue Tracking, and Containerization.

Moreover, in line with the identified user requirements, NESTLER AI platform is designed to host AI algorithms for:

- weather impact assessment;
- animal health surveillance;
- agricultural crop surveillance and management;
- pest infestation monitoring.

All these algorithms are based on IoT sensor data, drone data, camera data, microphone data and SEO data that can be saved on the platform via dedicated interfaces.

Moreover, a high-level system architecture of NESTLER project, including data sources and services, is presented.

1 Introduction

The delivery D4.1 is the first deliverable of the WP4. It contains reports of work already achieved and related the following NESTLER work package tasks:

- T4.1: AI algorithms for external weather impact assessment upon agriculture farming.
- T4.2: Automated monitoring services for crop yield quality, livestock wellbeing and insect population.
- T4.3: Economic risk assessment models for predicting the yield quality.
- T4.5: NESTLER platform architecture design and integration.

This document is organized into seven sections, as it follows:

- *Section 1* lists the NESTLER's work package tasks related to D4.1 and present an overview of the document's content;
- *Section 2* introduces the AI Federated Learning concept, and it describes some of its documented implementations (algorithms and frameworks) in a context where security and data privacy rules have to be enforced.
- *Section 3* describes NAIF which is the NESTLER AI framework aiming to allow a flexible deployment of the Federated Learning frameworks for the Federated Learning based model training to be implemented by NESTLER.
- *Section 4* describes the NESTLER integration framework and tools aiming to ensure a structured and flexible integration of all NESTLER's subcomponents.
- *Section 5* presents some of the AI applications developed by NESTLER partners and which are going to be tested and validated in the forthcoming months within the NAIF.
- *Section 6* describes NESTLER reference architecture as it has evolved from the beginning of the project guided by the system intended functionality and the identified user requirements.
- Finally, *section 7* concludes this report and draws the perspectives of works to come.

2 AI Federated Learning Framework

Within NESTLER, different types of data will be utilized in order to train AI models and extract helpful results. In many cases, these data are or will be distributed in various data repositories that are owned by different organizations, located in most cases in different countries. Most importantly, the data owners may be reluctant to offer these data for various reasons (i.e. confidentiality, competition, politics,…). As a solution, ***Federated Machine Learning (FML)*** has been identified as a reliable technique for training of ML models using distributed/decentralized data [1]. A group of distributed nodes can work together through a federation to collectively train a machine learning model without revealing their individual data to each other. In Federated Learning (FL), each node conducts model training independently on its own data and subsequently shares its model with a central "server" node, often referred to as the "Aggregator" in FL terminology. There are state-of-the-art FL frameworks available to facilitate federated training, whether in real-world or simulated environments, as elaborated in the section §2.2.10. Then we proceed by analyzing the NESTLER AI backend approach.

2.1 Federated ML aggregation algorithms

Federated ML allows a group of participants to collectively train a model by exchanging the model parameters instead of sharing the individual participant's data. This concept has been extensively researched in the literature with the goal to enhance the distributed training process and reduce costs. Federated Learning aggregation algorithms are the key mechanisms for updating the global model located on the central server during each communication round or at regular intervals. A groundbreaking work for global model aggregation is the ***Federated Averaging Algorithm (FedAvg)***. In the next sub-sections, *FedAvg* along with some alternatives are presented.

2.1.1 The Federated Averaging Algorithm

As a global model aggregation algorithm Federated Averaging (*FedAvg*) [2] is located and running in the central server when it obtains the client's updated model weights. This method employs gradient methods (*SGD*) which can be adapted to a federated learning setup. However, FedAvg performs more local computation and reduced communication compared to *SGD*. Furthermore, it reduces the communication cost by selecting a subset of clients during each training round and computing the *SGD* for these clients. The primary goal of *FedAvg* is to minimize the objective of the global model *w*, which is the weighted sum of the average of individual devices losses:

$$\min_{w} f(w), \quad f(w) \sum_{k=1}^{n} p_k F_k(w) \qquad (1)$$

where *n* is the total number of devices, $F_k(w)$ is the local objective function for the k_{th} device, while $p_k \geq 0$, specifying the relative impact of each device and $\sum_k p_k \sum_k p_k = 1$. The authors analyze *FedAvg* in-depth and demonstrate how partial client participation does not affect the learning process.

2.1.2 FedAvg alternatives

This sub-section introduces several alternatives of FedAvg. Each Federated Learning aggregation algorithm tries to tackle a specific challenge.

- **FedProx** [3] is an algorithm aimed to address statistical heterogeneity among client devices. Compared to FedAVG, *FedProx* exhibits superior performance on non-identically distributed data and demonstrates more robust convergence. The primary innovation in *FedProx* involves the incorporation of a proximal term on each client device to enhance the method stability, thereby limiting the impact of each local model updates on the global model. This proximal term offers a principled way for the server to account for heterogeneity associated with partial information. Similar to *FedAvg, FedProx* assigns equal weight to all devices during global aggregation, overlooking hardware heterogeneity.
- **FedPAQ** (Federated Learning Periodic Averaging and Quantization) [4] is a federated learning aggregation algorithm aiming to reduce communication costs. *FedPAQ* incorporates periodic averaging mechanism which means that models are trained and updated locally on the devices. These models are then periodically averaged at the server, with the frequency of averaging determined by a parameter linked to the number of local iterations. Similarly, to *FedAvg* only a portion of clients participates in each training iteration. Another notable aspect is that *FedPAQ* quantizes each nodes' updates before uploading them on the central server. This quantization technique significantly helps to reduce the communication overhead on the network.
- ***Turbo-Aggregate*** [5] aims to low communication expenses while simultaneously bolstering security. To achieve this, Turbo-Aggregate employs a multi-group circular approach for model aggregation. Specifically, clients are divided into multiple groups, and during each aggregation process, model updates are circulated among the groups in a circular fashion. According to the authors, this approach effectively diminishes the overhead associated with aggregation. Furthermore, Turbo-Aggregate introduces an additive secret sharing mechanism to safeguard the privacy of clients' data, thereby enhancing privacy preservation.
- ***HierFAVG*** [6] facilitates partial model aggregation across multiple edge servers with the goal of minimizing communication expenses. HierFAVG operates within a hierarchical client-edge-cloud architecture, where each server updates its own set of clients. Global aggregation occurs with edge-level aggregate models after a predefined number of model aggregations.
- **FedMA** [7] aims to tackle statistical heterogeneity in federated learning scenarios through a layer-wise learning approach. This method involves identifying and merging nodes with similar weights. What sets FedMA apart is its consideration of the permutation invariance of neurons in a neural network model prior to aggregation. Additionally, it employs a Bayesian non-parametric technique to adapt to changes in the global model's size. FedMA outperforms FedAvg in terms of both performance and communication efficiency. However, it should be noted that FedMA is primarily suitable for simple neural network architectures, such as fully connected feedforward networks.

In literature, there are also other approaches [8] [9] for different aggregation mechanisms which one can refer to if a more extended review is needed.

The following table provides a summary of the federate ML aggregation mechanism. As already mentioned within NESTLER we will utilize the FedAvg algorithm.

Table 1: A summary of the Federated ML aggregation mechanisms.

Federated Aggregation Method	Challenge to address	Summary
FedAvg	Statistical	FedAvg functions as a global model aggregation algorithm, operationalized on the central server upon receiving updated model weights from client devices. It leverages basic gradient methods, such as Stochastic Gradient Descent (SGD). To diminish communication costs, FedAvg opts to select a subset of clients during each training round, computing the SGD specifically for this chosen group.
FedProx	Statistical	FedProx outperforms FedAvg when dealing with non-identically distributed data and exhibits enhanced convergence robustness. It achieves this by incorporating a proximal term on each client device, which bolsters method stability. In FedProx, equal weighting is applied to all devices during global aggregation, with no consideration for hardware heterogeneity.
FedPAQ	Communication	FedPAQ incorporates periodic averaging from clients to the server, controlled by a parameter corresponding to the number of local iterations. Like FedAvg, only a subset of clients is involved in each training iteration. Furthermore, FedPAQ quantizes the updates from each node before transmitting them to the central server.
Turbo-Aggregate	Communication and Privacy	Turbo-Aggregate utilizes a multi-group circular approach for model aggregation. Clients are divided into multiple groups, and during each aggregation round, model updates are circulated in a circular fashion among these groups. To enhance privacy preservation, Turbo-Aggregate introduces an additive secret sharing mechanism that safeguards the privacy of clients' data.
HierFAVG	Communication	HierFAVG enables multiple edge servers to conduct partial model aggregation with the goal of minimizing communication expenses. This approach is rooted in a hierarchical client-edge-cloud architecture, where each server updates its respective clients. The global aggregation occurs using aggregate models at the edge level and takes place after a predefined number of model aggregations.
FedMA	Statistical	FedMA incorporates a layer-wise learning scheme that involves matching and merging nodes with similar weights. Additionally, it takes into account the permutation invariance of neurons in a neural network model before conducting aggregation. To adapt to changes in the global model size, FedMA employs a Bayesian non-parametric method. As a result, FedMA exhibits superior performance and enhanced communication efficiency compared to FedAvg.

2.2 Federate Machine Learning (FML) Frameworks

Numerous open-source Federated Machine Learning (FML) frameworks have emerged to enable distributed learning on decentralized data while also bolstering privacy and security. Google proposed ***TensorFlow Federated*** [10] platform designed for machine learning and other computations on decentralized data. Another notable open-source federated learning framework is ***PySyft*** [11], which originated from OpenMined. PySyft is well-suited for research in Federated Learning and provides users with the capability to conduct private and secure deep learning tasks. It is also integrated into PyGrid, a peer-to-peer platform designed for federated learning and data privacy, enabling private statistical analysis on sensitive datasets and federated learning across multiple organizations' datasets. WeBank's AI department introduced ***FATE*** (Federated AI Technology Enabler) [12] yet another open-source framework that supports various Federated Learning architectures and enables secure computation of numerous machine learning algorithms. FATE is primarily geared towards industrial applications and enterprise solutions. ***Flower*** [13] tands out as a user-friendly open-source federated learning framework that is agnostic to machine learning frameworks. It offers higher-level abstractions, making it easier for researchers to experiment and build upon a dependable stack. Another promising open-source Federated Learning framework is ***Sherpa.ai*** [14] which integrates federated learning with differential privacy. Sherpa.ai combines federated learning with differential privacy principles, offering a solution for machine learning applications in a federated manner. ***FedML*** [15] yet another open-source federated learning framework, serves as both a framework and benchmarking tool for federated machine learning. FedML supports three computing paradigms: on-device training for edge devices, distributed computing, and single-machine simulation. Its generic API design and comprehensive reference baseline implementations promote diverse algorithmic research in the federated learning domain.

Another well-known open-source federated learning framework is the ***PaddleFL*** [16], that allows easy replication and comparison of different federated learning algorithms. It is designed for deployment in large-scale scenarios. ***Leaf*** [17] s a modular benchmarking framework for federated learning with a wide range of applications, including federated learning, multi-task learning, meta-learning, and on-device learning. ***OpenFL*** [18] is another open-source federated learning framework that focuses on data-private collaborative learning within the federated learning paradigm. It works seamlessly with machine learning pipelines built on top of TensorFlow and PyTorch and can be customized to support other machine learning and deep learning frameworks. ***NV FLARE*** (NVIDIA Federated Learning Application Runtime Environment) [19] is one of the most recent open-source federated learning frameworks. It provides developers with the tools to build secure and privacy-preserving federated systems where nodes can share model information while guarding against potential malicious attacks. NV FLARE enables participants from different locations to join the federated learning process and train the model with their own data.

In the following subsections, a more detailed analysis of each of these federated learning frameworks is provided. In the next section, a comprehensive comparative analysis of these frameworks in the context of NESTLER's objectives and use cases is presented.

2.2.1 TensorFlow Federated (TFF)

TensorFlow Federated (TFF) [10] is a federated learning framework designed for conducting research in the field of federated learning. It achieves this by simulating federated computations on realistic proxy datasets. Within TensorFlow Federated, you'll find TensorFlow Privacy, a Python library tailored for implementing privacy techniques in machine learning model training. Furthermore, there is a

wealth of useful tutorials available, covering topics such as image classification and text analysis, making it seamlessly integratable with existing TensorFlow-based machine learning models. TensorFlow Federated offers two distinct APIs: the Federated Learning API, which provides a range of high-level interfaces, and the Federated Core API, which offers low-level interfaces for creating customized federated algorithms, enabling complex computations involving multiple participants. Figure 1 presents the architecture overview of *TFF*. *TFF* implements *FedAvg* and Federated Stochastic Gradient Descent (*FedSGD*) algorithms.

Regarding the limitations of *TFF*, firstly, it lacks support for a true federated mode, making it unsuitable for conducting real-life experiments or commercial applications. Another drawback is its lack of support for vertical and hybrid data splitting methods. In terms of privacy-preserving mechanisms, TFF Privacy does incorporate Private Aggregation of Teacher Ensembles (PATE), which is discussed in detail in section2.3.4. However, it's worth noting that TFF benefits from a substantial community of contributors.

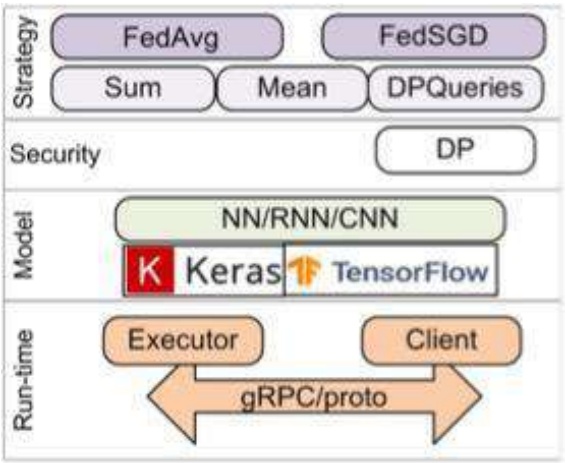


Figure 1: Architecture Overview of TensorFlow Federated (TFF) [10]

2.2.2 PySyft + PyGrid

PySyft [11] is an open-source federated learning tool, licensed under MIT, designed for secure deep learning applications. It achieves data privacy by employing differential privacy and Secure Multi-Party Computation (MPC) techniques, effectively decoupling private data from the model training process. PySyft stands out as one of the leading federated learning frameworks, boasting a large and active support community. While PySyft is exceptionally useful for research purposes and prototyping, it leans more towards simulation than serving as a production-ready solution. Key features of PySyft include its:

- Compatibility with popular Deep Learning frameworks like TensorFlow and PyTorch
- A low-level Federated Learning implementation that aids in project development and debugging
- Robust privacy mechanisms, primarily centered around secure MPC using Homomorphic encryption. This encryption enables computations on ciphertexts, a significant advantage in developing Federated Learning applications.

However, PySyft does have a notable limitation: it operates effectively only in simulation mode. To enable federated mode, it requires integration with another open-source project called PyGrid.

2.2.3 FATE

FATE (Federated AI Technology Enabler) [12] is an open-source initiative spearheaded by WeBank's AI Department. Its primary objective is to furnish a secure computing framework tailored to bolster federated machine learning applications, with a particular focus on regression analysis, decision trees, and transfer learning. FATE is meticulously crafted for industrial use cases, and its key attributes include:

- The framework incorporates commonly utilized horizontal and vertical federated algorithms for data engineering and machine learning. It also offers a workflow engine that enables the development of customizable end-to-end machine learning tasks.
- FATE provides its own distributed computing, transmission, and storage engine, which is designed to handle large-scale applications effectively.
- It offers a high-level interface powered by custom scripts and boasts support for various Federated Learning (FL) algorithms.
- FATE prioritizes security and includes support for Multi-party Computation mechanisms and encryption techniques such as RSA, Paillier, and homomorphic encryption.
- FATE supports both simulation and federated modes through Kubernetes clusterization for versatility in deployment.

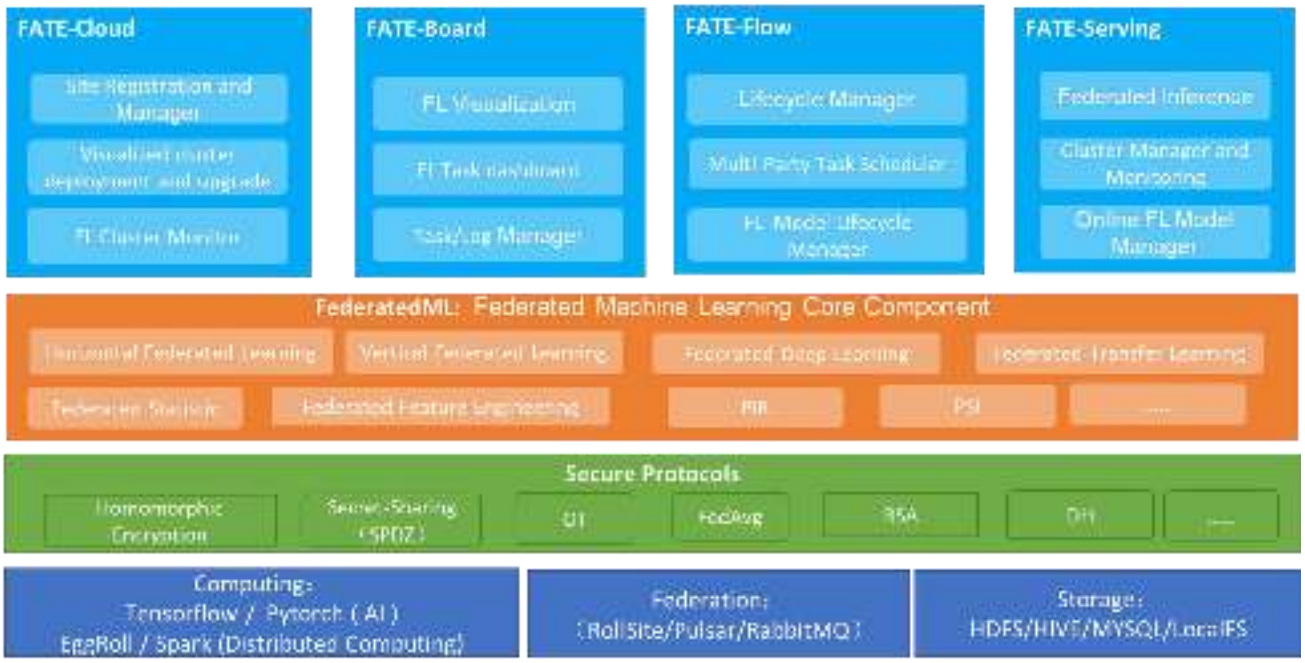


Figure 2: Architecture overview of FATE framework [12].

FATE's architecture is presented in Figure 2. Following, the four independent platforms are listed:

- FATE-Cloud encompasses a cloud manager responsible for overseeing federated site management and FATE Manager, serving as a client management terminal for individual sites. It facilitates tasks such as federated site registration, management, automated cluster deployment and upgrades, cluster monitoring, and permission control, among other core functions.

- FATE-Board serves as a visualization tool tailored for exploring federated learning models in-depth, making model understanding more accessible and effective.
- FATE-Flow essentially represents FATE's workflow system. It handles the scheduling and management of the entire lifecycle, enabling the creation of end-to-end pipelines for federated learning production services.
- FATE-Serving is a high-performance and scalable online model-serving service designed to support vertical federated learning use cases.

In summary, FATE stands out as a meticulously crafted Federated Learning framework with substantial potential, particularly for production environments. However, it exhibits high resource requirements, rendering it less suitable for IoT devices and edge computing scenarios. Another notable limitation of FATE is its current lack of support for comprehensive deep learning model training. Nevertheless, the most significant challenge is the absence of detailed English-language documentation for FATE's custom language (DSL), which can pose difficulties for users.

2.2.4 Flower

Flower [13] is an open-source federated learning framework that caters to heterogeneous environments, accommodating a range from IoT devices and mobile phones to scaling up for thousands of clients. Flower's architectural design simplifies the integration of workflows from existing ML applications, regardless of the underlying ML/DL framework (such as PyTorch [20], TensorFlow, etc.), with minimal performance impact. Moreover, it enhances research endeavors by virtue of its flexibility and interoperability. Flower is developed with the following design objectives in mind:

- **ML framework-agnostic** – This is a significant advantage as ML frameworks continually evolve, and any alterations to running Federated Learning applications could potentially result in severe software issues. Flower is compatible with nearly all machine learning frameworks.
- **Client agnostic** – Participant devices typically exhibit substantial structural heterogeneity. Addressing this heterogeneity is critical, and Flower maintains interoperability across various operating systems, programming languages, and hardware characteristics.
- **Expandable** – Many Federated Learning frameworks lack expandability, which limits their adaptability to emerging research efforts. Flower is designed to be expandable, facilitating the adoption of newly proposed methods and accommodating evolving research needs.
- **Accessible** – Most Federated Learning frameworks can substantially increase engineering complexity when customizing ML applications for federated execution. Flower simplifies the process, allowing users to easily federate existing ML pipelines.
- **Scalable** – In real-world scenarios, cross-device Federated Learning often involves a multitude of clients. Flower is well-equipped to scale efficiently to accommodate a large number of clients.

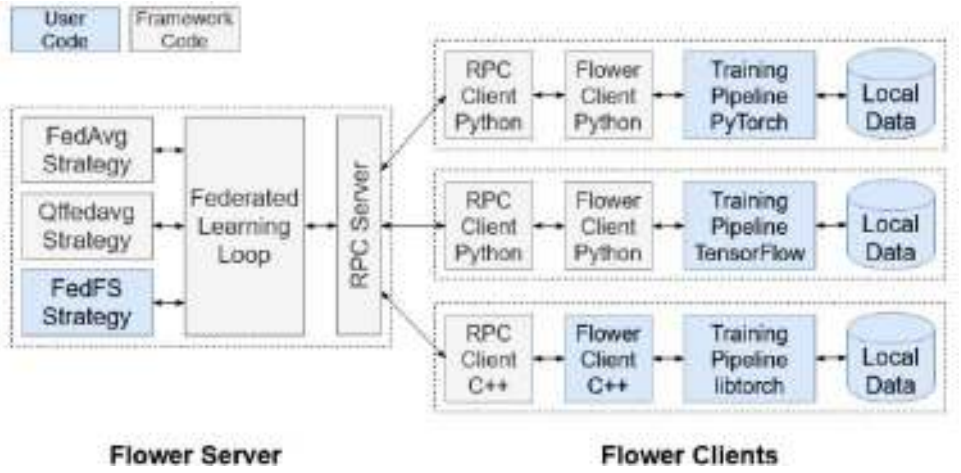


Figure 3: Flower core framework architecture [13]

In Figure 3, we present the fundamental architecture of Flower's core framework, meticulously designed to scale efficiently as the workload intensifies. As illustrated in the figure, Flower is divided into two main components: Server and Clients. The server-side comprises three primary elements: the Federated Learning loop, the RPC server (the connection server) responsible for transmitting and receiving Flower Protocol messages, and a user-customizable Strategy.

Clients initially establish a connection with the RPC server, while the Federated Learning loop serves as the central orchestrator throughout the entire federated learning process.

2.2.5 Sherpa.ai

Sherpa.ai [14] is an open-source federated learning framework. It comprises seven distinct modules, each designed to encapsulate specific functionalities corresponding to key elements within a Federated Learning (FL) environment, as shown in Figure 4, which are the following:

- Data base: This module is responsible for data retrieval and processing based on the selected database.
- Data distribution: Within this module, the federated distribution of data among participants is managed.
- private: This module encompasses various interfaces, including the node interface representing a client's core element, along with other interfaces for modifying federated data distribution.
- Learning approach: This module establishes the software's architecture for developing federated aggregation operators.
- model: This module defines the machine learning model employed in the framework.
- Differential privacy: This module is tasked with ensuring differential privacy for all participants in the system.

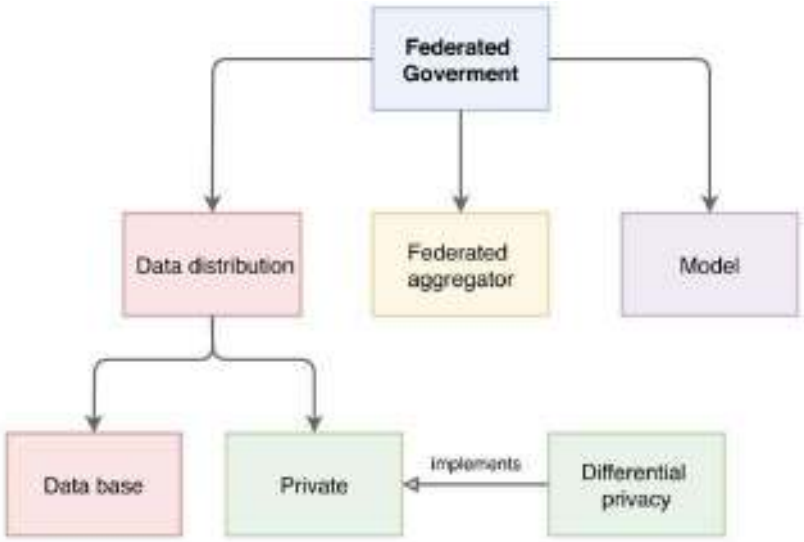


Figure 4: Overview of the Sherpa.ai module architecture [14]

Sherpa.ai boasts robust privacy-preserving capabilities and user-friendliness; however, it falls short in comparison to NESTLER in certain aspects. Notably, it can only operate in simulation mode and has limited applicability across various scenarios, which are essential features available in NESTLER.

2.2.6 FedML

FedML [15] is open-source library and benchmark tool for Federated Learning projects offer a comprehensive toolkit for facilitating the development of Federated Learning models and serves as a reliable performance comparison tool across a wide range of computing configurations. FedML incorporates the following key design elements:

- FedML offers support for three distinct computing environments:
  - On-device training tailored for edge computing, including mobile phones and IoT devices.
  - Distributed computing, adhering to federated learning principles.
  - Single-machine simulation for testing and development purposes.
- FedML adopts a worker/client-oriented programming scheme, facilitating diverse network topologies, flexible information exchange, and accommodating various training workflows.
- Standardized Federated Learning algorithms are provided by FedML, simplifying the learning curve for users and serving as a baseline for comparing newly developed algorithms.
- FedML also supplies standardized benchmarks, complete with specific evaluation metrics, alongside a wide array of synthetic and real non-IID datasets to support comparison and benchmarking endeavors.

Figure 5 presents the architecture overview of *FedML*, which consists of two key components, *FedML-API* and *FedML-core* which represents high-level API and low-level API, respectively.

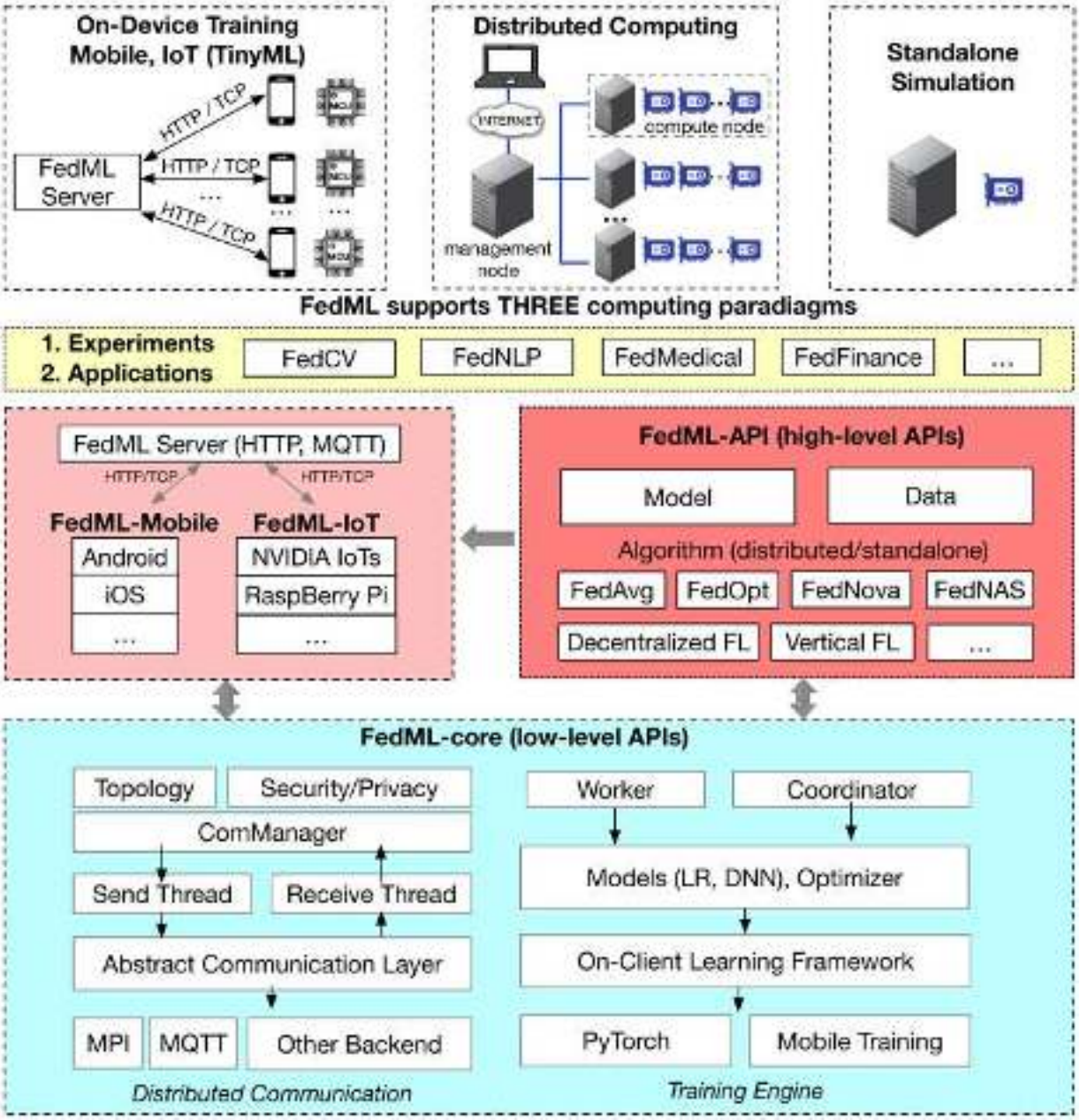


Figure 5: Architecture Overview of FedML and its components [15]

FedML-core comprises two distinct components: the distributed communication component, responsible for low-level communication among workers/clients, and the training engine. On top of FedML-core, we have the FedML-API, which is well-suited for the implementation of new federated learning algorithms, utilizing a client-oriented programming interface. Additionally, FedML-API introduces a machine learning system practice that separates the development of models, datasets, and algorithms, promoting the ease of implementation reuse.

FedML offers specialized sub-tools like FedML-Mobile and FedML-IoT, designed to cater to various real-world scenarios. However, a notable limitation of FedML is its current lack of privacy-preserving mechanisms.

2.2.7 PaddleFL

PaddleFL [16] is an open-source Federated Learning framework capable of handling both horizontally and vertically partitioned data. It's well-suited for deployment in large-scale clusters and offers support for a range of Federated Learning strategies tailored to diverse application scenarios, including but not limited to Computer Vision and Natural Language Processing.

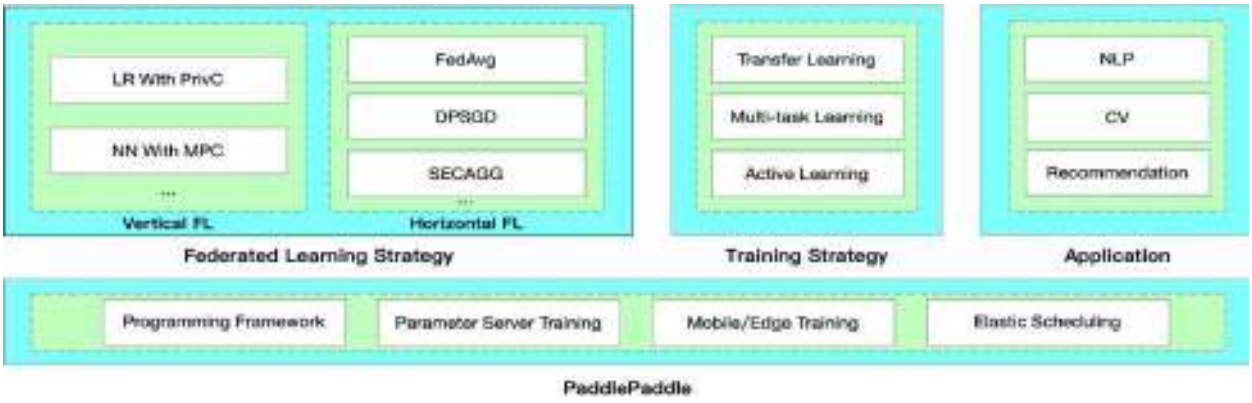


Figure 6: Architecture overview of PaddlePaddle deep learning platform [16]

PaddleFL is based on *PaddlePaddle* (PArallel Distributed Deep LEarning) (Figure 6), is a Deep learning platform specifically designed as an industrial-grade platform incorporating advanced technologies. This comprehensive platform encompasses deep learning frameworks, fundamental model libraries, development kits, and various components. PaddleFL consists of two primary components:

- Data-Parallel – This component empowers data owners to define their Federated Learning tasks using common horizontal strategies like FedAvg, among others.
- Federated Learning with MPC – It facilitates secure training and prediction on both vertical and horizontal datasets while also supporting transfer Federated Learning.

However, one notable drawback is that PaddleFL relies on a relatively lesser-known Deep Learning platform (PaddlePaddle) with limited community contribution and insufficient documentation.

2.2.8 LEAF

Leaf [17] is an open-source modular benchmarking framework for Federated Learning. It encompasses various open-source datasets, an evaluation framework, and a collection of Federated Learning algorithm implementations. LEAF is distinguished by three primary characteristics:

- Facilitating reproducible scientific research.
- Offering detailed and granular metrics.
- Providing a modular framework for customization.

Similar to PaddleFL, LEAF operates with a relatively small team of contributors. Its primary emphasis is on benchmarking Federated Learning settings, although it does not offer benchmarking tools for privacy-preserving techniques

2.2.9 OpenFL

OpenFL [18] is an open-source framework that facilitates training machine learning models using the principles of Federated Learning. It offers support for both TensorFlow and PyTorch, with the flexibility to easily extend its compatibility to other machine learning and deep learning frameworks. OpenFL is intentionally designed to be versatile across different project domains, making it applicable in a variety of contexts, including healthcare and IoT applications. It comprises two fundamental components: the Aggregator and the Collaborator. A Collaborator node serves as a client, housing the dataset owned by a participant on which the learning model is applied. In contrast, an Aggregator node functions as a computational unit responsible for receiving and amalgamating locally fine-tuned model updates from collaborators into a new global model.

After installing OpenFL on all computation nodes within the federated system and ensuring that each client has received the PKI certificate, the workspace is distributed to every node. OpenFL generates a valid Public Key Infrastructure certificate, referred to as PKI, which is created and signed by a trusted authority. However, it's important to note that this certificate is not suitable for production-level tasks. The overarching design of OpenFL is built around the concept of the Federated Learning Plan, which plays a pivotal role in configuring and executing the project's workflow. Specifically, the Federated Learning Plan is a YAML file that outlines the tasks and essential parameters necessary for coordinating and executing a federated task. It encompasses details such as the collaborators, aggregator, connections, models, data, and any other parameters that delineate the course of the training process. In Figure 7, the backend (in blue) connects the collaborator with the aggregator through a TLS connection using the PKI certificate. Aggregator's backend (in blue) sends remote procedure calls to the collaborator and receives model and metric updates for aggregation.

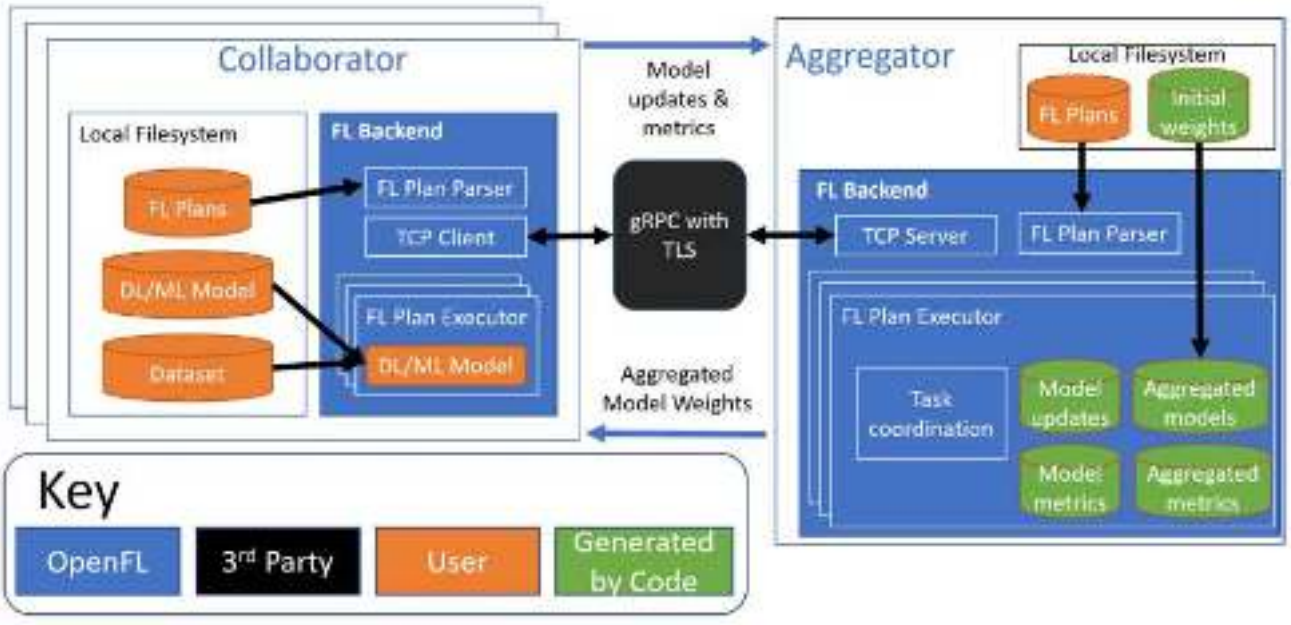


Figure 7: A high-level architectural overview of OpenFL with its components [18]

2.2.10NVIDIA FLARE

NV FLARE (NVIDIA Federated Learning Application Runtime Environment) [19] is an open-source Federated Learning framework designed to empower researchers and data scientists to adapt their existing workflows, whether implemented in PyTorch, TensorFlow, or even NumPy, and deploy them

within a federated environment. The primary objective of NV FLARE is to enable developers to construct a secure and privacy-preserving federated system. This framework facilitates the secure sharing of model information among nodes while guarding against potential malicious attacks. It facilitates the participation of all contributors from various locations in the Federated Learning process, allowing them to collectively train the model using their respective datasets.

NV FLARE primarily comprises the main node, which serves as the server, and the federated nodes, acting as clients. These nodes engage in a continuous cycle of communication. Specifically, the server takes on the responsibility of disseminating tasks to the clients, such as training and validation. Once the clients have completed their assigned tasks, they report the results back to the server, where they are aggregated. Importantly, all tasks can be configured with filters that pertain to privacy and security mechanisms. Two basic entities manage the federated learning (Figure 8). On one hand, the *Controller* runs on the FL server and controls or coordinates the FL clients to get a job done. On the other hand, the *Worker* runs on FL clients and can perform tasks.

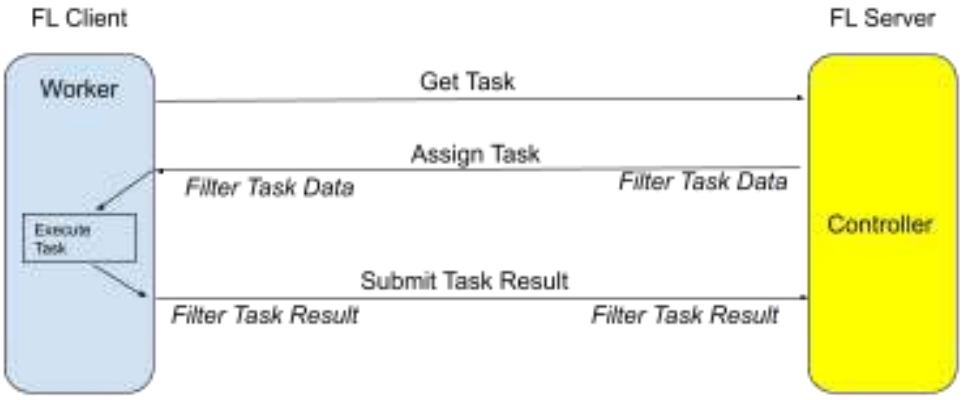


Figure 8: NVIDIA Flare Controller/Worker interactions [19]

In more technical detail, the system involves the creation of two configuration JSON files. The first file, named "config_fed_server.json," is dedicated to the server and encompasses critical details about the entire implementation process. It can specify elements such as the model architecture, the methodology for weight aggregation, and even the number of rounds to be executed. The second configuration file, "config_fed_client.json," is tailored for the clients and serves to define the file paths required for deploying tasks within the system. Communication between the server and the clients is facilitated through a Shareable object, which, from a technical standpoint, is implemented as a Python dictionary. This dictionary carries essential meta-information, including the peer identity name, the task name to be executed by the client, ML model weights, and more. The execution of tasks on the clients is orchestrated by the Executor class. This class is responsible for retrieving the task name (e.g., train or submit model) and all the necessary information from the Shareable object, subsequently executing the designated task. Figure 9 presents the FL process performed by NVIDIA FLARE. Initially, the server initiates the creation of the training model architecture and transmits it to the clients. Upon receiving the model, the clients remain in a waiting state, anticipating instructions from the server regarding the task to be executed. If the initial task is "training," each client commences the training process using its respective dataset. Upon completing the training phase, each client transmits their model weights back to the server. The server then aggregates these weights from the clients and subsequently disseminates the newly aggregated weights to all clients. This iterative process is carried out for each round. The final round of this iterative cycle involves a validation process. Both the server and the clients perform validation procedures on the final model using their respective datasets. NV

FLARE offers 'admin' packages to lead scientists and administrators, empowering them to oversee the entire federated process. These packages enable various control actions, including job submissions for deploying applications, status checks, and the ability to abort or shut down ongoing training processes. Specifically, the admin packages include key and certificate files, facilitating secure connection and authentication with the server. Administration tasks can be executed via an included command prompt or programmatically through the FLAdminAPI. [21].

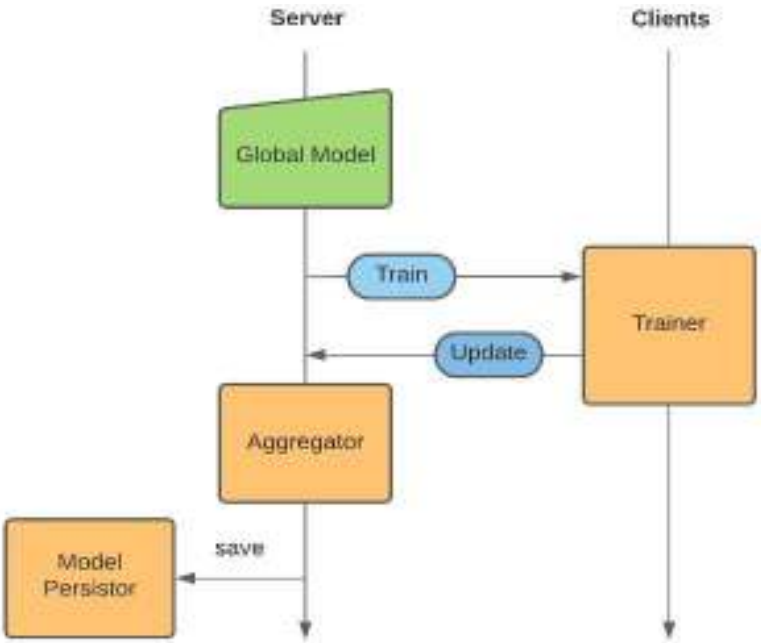


Figure 9: FL Training Workflow.

NV FLARE offers a range of privacy-preserving mechanisms, including percentile privacy and homomorphic encryption, among others. These supported privacy-preserving mechanisms can be implemented as filters during the transmission of information between peers. Below, we provide a detailed description of these mechanisms.

2.3 Federated Learning (FL) algorithms privacy

Various Federated Learning (FL) methods facilitate machine learning without the need to share personal or proprietary data. However, it's important to note that the absence of data sharing does not guarantee protection against potential extraction of personal information from the data. Here's how the process typically unfolds:

1. A shared model is trained on a user's device using their confidential data.
2. The trained parameters, which include AI model weights, are transmitted to a central server.
3. Through an aggregation mechanism, the global model is constructed using these parameters.

During the transfer of the model, there exists the possibility for an adversary to extract information related to private data from these trained parameters. For instance, a recurrent neural network trained on user data might be capable of extracting sensitive text patterns, such as credit card numbers. To address this concern, it becomes imperative to implement additional mechanisms that safeguard against data disclosure from potential attack strategies [22].

2.3.1 Differential Privacy (DP)

Among the various proposed methods, ***Differential Privacy (DP)*** is a method that randomizes part of the mechanism's behaviour to provide privacy [23]. The motivation for introducing randomness, whether in the form of Laplacian or Gaussian noise, into a learning algorithm lies in the objective of preventing the disclosure of data patterns or insights associated with both the model and its learned parameters, as well as the training data. This approach serves as a privacy safeguard against a broad spectrum of attacks, including differencing attacks and linkage attacks. It's important to note that DP, when applied appropriately, can provide privacy without necessarily compromising the accuracy of the learning algorithm. While there have been studies highlighting instances where DP may lead to some sacrifices in terms of accuracy, it is by no means a universal outcome. The effectiveness of DP in preserving both privacy and accuracy depends on various factors, including the specific implementation and settings employed in a given context. [22].

Differential privacy indeed offers robust guarantees for privacy protection. In contrast, in the absence of randomness, potential adversaries could extract insights related to the parameters essential for the learning and convergence processes on datasets or discern the probabilities governing the selection of parameters within a set of possible learning parameters for a given dataset. The incorporation of differential privacy techniques serves to mitigate these vulnerabilities. Generally, differential privacy may be divided into *Local Differential Privacy (LDP)* [24] and Global Differential Privacy (GDP) [25].

Local Differential Privacy offers a means to perform statistical computations while safeguarding the privacy of each individual user. This approach doesn't rely on placing trust in a central authority or a third party, as noise is added to the individual inputs at the local level. To illustrate, let's consider the local nodes in the diagram provided in Figure 10 as the context of Local Differential Privacy. Noise distributions are added in each one of these nodes, ending up to the untrusted aggregator. Local differential privacy ensures that no trusted party is required since the individuals are responsible for adding noise to their own data before they share it.

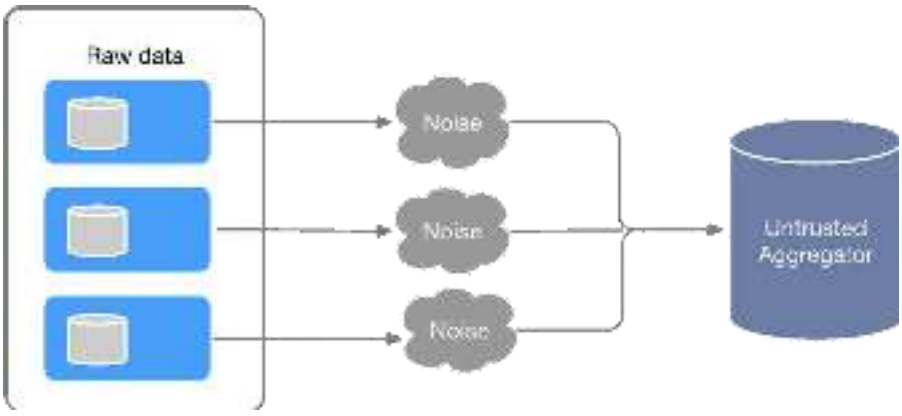


Figure 10: Local Differential Privacy Scheme

Global Differential Privacy techniques involve the presence of a central aggregator, often referred to as a trusted curator, which has direct access to the raw data. In this scenario, each user sends their data to the aggregator node without introducing any noise. Subsequently, the aggregator processes the incoming data using a differentially private mechanism, which typically entails the addition of noise, either Laplacian or Gaussian. When an untrusted querier poses a specific query to the trusted aggregator node, an answer is provided. However, it is important to note that this answer is designed

in such a way that it is mathematically impossible to reverse-engineer and deduce the precise details of the private raw data. This protective mechanism ensures that the privacy of the individual data contributors is preserved while still allowing for query responses. Figure 11 overviews the general scheme of a global differential privacy example.

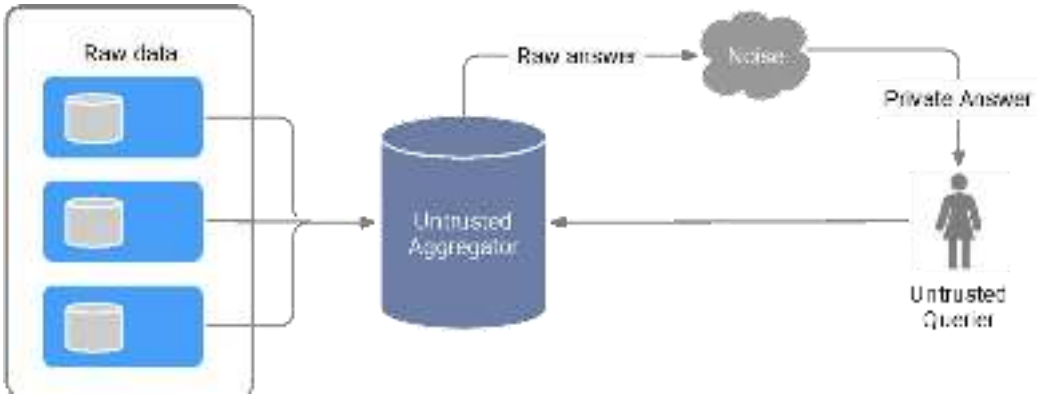


Figure 11: Global Differential Privacy Scheme

Another approach involves the utilization of a Differentially Private Stochastic Gradient Descent (DP-SGD) algorithm [26], which is designed to regulate the impact of the training data during the optimization operation, such as Gradient Descent (GD). In the DP-SGD algorithm, for each step of the optimization process, the gradient is computed based on a random subset of the data. The algorithm then calculates the clipped norm of each gradient, computes the average, adds noise to protect privacy, and subsequently takes a step in the opposite direction of this Stochastic Gradient Descent (SGD). It's worth noting that the execution of DP-SGD introduces a significant runtime overhead. To address this concern, various improvements have been proposed, including techniques like vectorization, just-in-time compilation (accelerated linear algebra), and static graph optimization, which have the potential to reduce this overhead by up to fifty times. [27]. Differential privacy in principle needs many clients to be effective and normally many clients are present in cross-device FL scenarios. Nevertheless, it is noticeable that clients tend to dropout after participating in even one training round, which creates a robustness problem.

2.3.2 Secure Multiparty Computation (SMC)

Another Federated Machine Learning (FML) technique worth mentioning is Secure Multiparty Computation (SMC) [28], which is a well-defined cryptographic-based approach that enables multiple mutually suspicious parties to jointly compute a function, preserving the privacy of their input data. In the context of SMC, the function to be jointly computed can be the ML training function, such as the model's loss function, or even the model itself during inference. However, applying SMC in a large-scale distributed system comes with the challenge of communication overhead, which increases significantly with the number of participating parties. Secure Multiparty Computation models offer security verification within well-defined simulation frameworks, guaranteeing complete zero-proof knowledge [29]. Specifically, universally verifiable Zero-Knowledge Proofs (ZKPs) are employed to ensure computation integrity and verify that encrypted data falls within the appropriate predefined ranges. Nonetheless, achieving this desired property can involve complex computations that may not be highly efficient in some scenarios. In several use cases, partial knowledge disclosure is considered acceptable, provided that pre-defined security guarantees are upheld. For example, an SMC framework

for training machine learning models in linear regression, logistic regression, and neural network training might utilize a stochastic gradient descent method with two servers and semi-honest assumptions [30]. Another approach involves Multi-Party Computation (MPC) protocols for machine learning model training, where sensitive attributes are encrypted [31]. This approach allows for the training and verification of a fair model without revealing sensitive attributes. As a result, these state-of-the-art approaches ensure privacy, even in the presence of semi-honest or malicious assumptions. However, practical systems often prioritize simplicity and pragmatism while maintaining necessary accuracy.

2.3.3 Homomorphic Encryption (HE)

Homomorphic encryption algorithms [32] enhance the security of the learning process by enabling mathematical operations to be performed on encrypted data. This property is highly valuable and finds applications in various contexts. Specifically, an encryption scheme is considered homomorphic when standard mathematical operations can be directly applied to the ciphertext data. The result of these operations, when decrypted, is equivalent to performing analogous operations on the original unencrypted data. There are different types of homomorphic encryption, including partially homomorphic, somewhat homomorphic, leveled fully homomorphic, and fully homomorphic encryption. In the context of machine learning methods, homomorphic encryption can be applied when training or inference is carried out directly on encrypted data (ciphertexts). However, a significant challenge arises in scenarios where extensive mathematical operations are performed in the ciphertext space. The properties of homomorphic encryption schemes encounter certain limitations, particularly regarding encryption performance.

For example, in Figure 12, we can observe the application of the sum operator on two values, A and B. Initially, both of these values are private data, and they are encrypted separately using a key named P. Next, we apply the sum operator to the encrypted values of A and B. The result of this encrypted sum can be decrypted using the key P, and it will match the original sum of the non-encrypted values or parameters A and B [33]. However, in scenarios where more complex mathematical computations or functions need to be executed within the ciphertext space, the remarkable properties of homomorphic encryption schemes face limitations related to encryption performance.

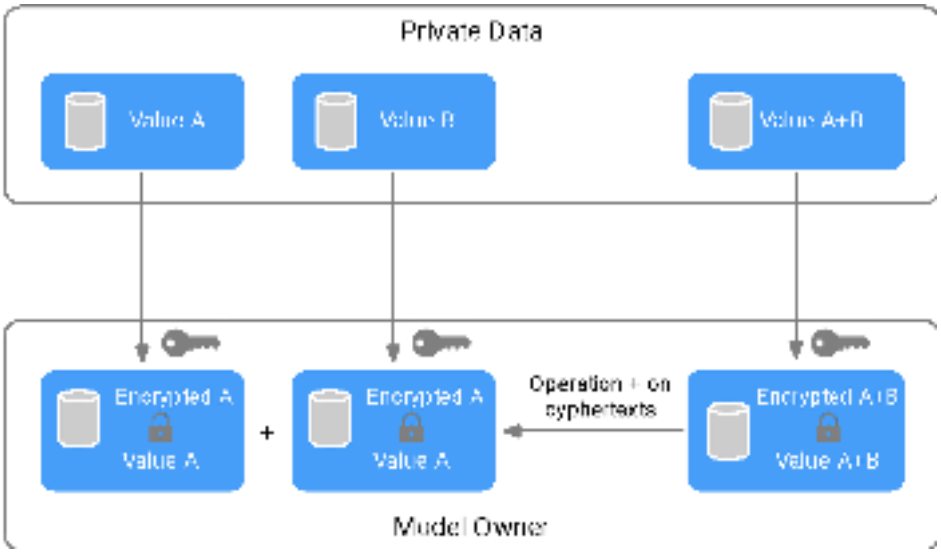


Figure 12: Homomorphic Encryption Example.

Numerous techniques have been introduced in the literature to address these limitations. These approaches aim to leverage statistical techniques to align with the properties of homomorphic computation. Additionally, they involve quantifying reasonable approximations in scenarios where traditional methods cannot be implemented homomorphically [34].

2.3.4 Private Aggregation of Teacher Ensembles (PATE)

A popular approach is the Private Aggregation of Teacher Ensembles (PATE) [35], which presents a teachers-student scheme [36]. Teachers are basically, the unpublished models trained on disjoint partitioned datasets with sensitive data. In Figure 13 PATE's general architecture is presented.

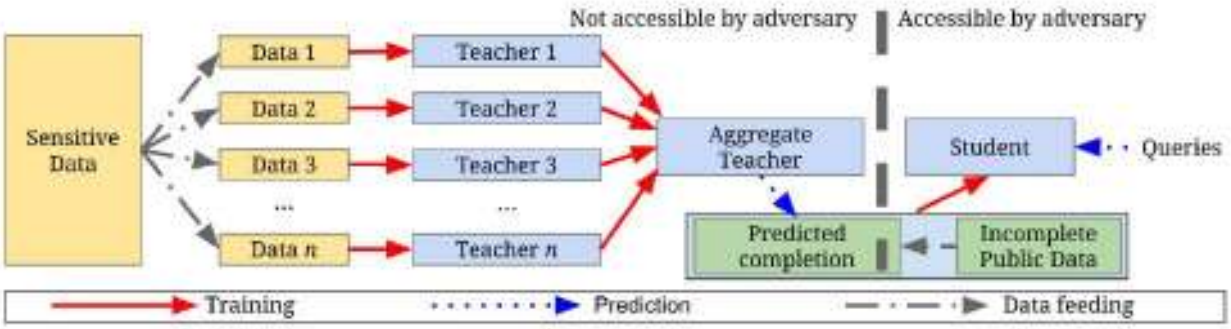


Figure 13: PATE approach diagram [35]

A student cannot directly access the teacher's data or parameters to learn and predict outputs. Instead, it relies on a noisy voting mechanism among all the teachers. The student model is trained semi-supervisedly using Generative Adversarial Networks (GANs). The GAN framework consists of two machine learning models: a generator, which produces samples from the data distribution, and a discriminator, which is trained to differentiate between fake samples (generated by the generator) and real samples from the data distribution.

To maintain privacy and limit the student's access to the teachers, Differential Privacy (DP) is employed. DP is generally defined using pairs of adjacent inputs, and Privacy Amplification by Teacher Ensembling (PATE) introduces a function to calculate privacy loss and the privacy loss random variable based on DP. To ensure privacy, the privacy cost associated with each outcome queried by the student must be bounded. The total cost of training the student is determined by applying the strong composition theorem.

To more efficiently track the privacy cost, PATE utilizes a technique known as the moments accountant, which bounds the privacy cost, especially when the topmost vote has a large quorum. In contrast to typical machine learning techniques, PATE divides the data into disjoint sets and trains a specific model for each set. An aggregated teacher then queries each individual teacher for predictions and combines their outputs into a single prediction. This aggregation mechanism guarantees privacy.

However, in cases where two classes have similar vote counts, disagreements among the teachers may reveal private information. To address this, PATE introduces random Laplacian noise (LNMax) to the vote counts and outputs the noisiest votes from the teacher's aggregator.

As an improvement in the teacher aggregator mechanism [36], it is suggested to replace LNMax with GNMax, a Gaussian-based noise distribution. GNMax is more concentrated than the Laplace distribution, and this modification enhances the utility of aggregation, especially when there is a large number of teachers. Additionally, it reduces the amount of noise required to achieve the desired level of privacy cost per student query.

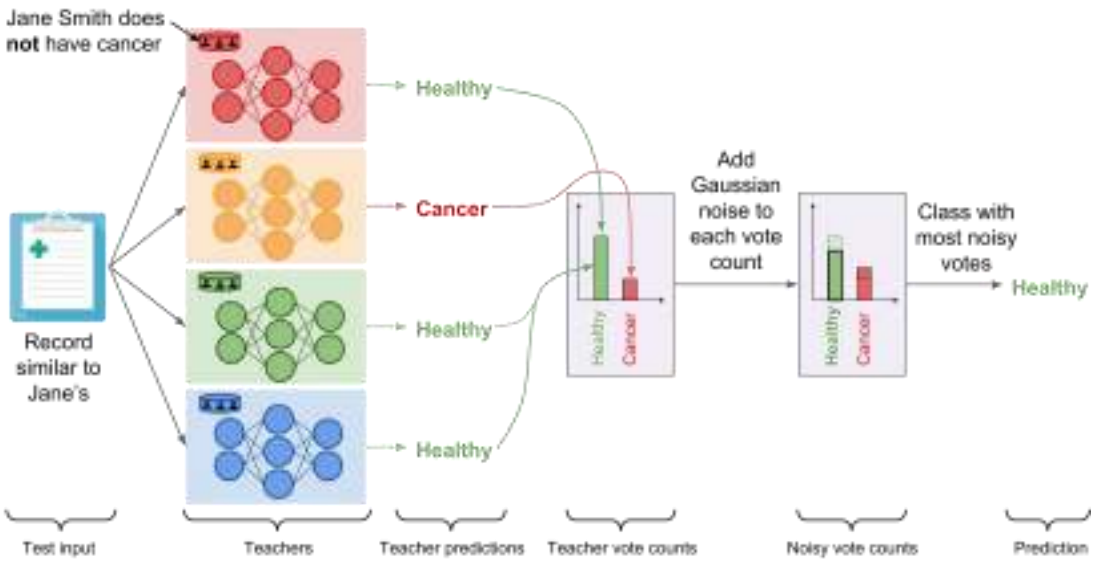


Figure 14: A detailed overview of PATE aggregation mechanism

In Figure 14 a detailed overview of how the aggregation mechanism works is presented while it is clear that preserving privacy using PATE does not affect the training process and results.

2.4 Comparison of Federated Learning frameworks

Based on the in-depth analysis provided in the preceding sections for both Federated Learning methods/tools and privacy-preserving approaches, a comparative analysis of Federated Learning frameworks is now presented in this section. The comparison refers to the FL frameworks analyzed in §2.2 and for which the main benefits and drawbacks are briefly presented in Table 2.

Table 2: Main pros and cons of the Federated Learning frameworks.

FL Framework	Main Pros	Main Cons
TensorFlow Federated	1. It seamlessly integrates with pre-existing TensorFlow ML models. 2. Its familiarity makes it user-friendly and easy to use.	1. It can only be utilized in simulation mode, lacking support for the federated operation mode. 2. Data for training cannot be loaded directly from the remote worker but must be partitioned and transferred through the central server.
PySyft & PyGrid	1. Easy to use 2. Largest community of contributors among the FL frameworks	1. PySyft is only for 1 server and 1 client 2. Run only in simulation mode 3. Need of PyGrid in order to develop real FL scenarios

FATE	1. Production Ready 2. High-Level interface 3. Provides many FL algorithms 4. Containerized - Kubernetes support	1. No differential privacy algorithms 2. Poorly documented domain-specific language 3. No core API 4. Doesn't use GPUs for training 5. Not well suited for the purposes of the NESTLER
Flower	1. Offers a template API that facilitates the transformation of ML pipelines into FL. 2. Simplifies development and is compatible with various ML frameworks. 3. Can handle a larger number of clients. 4. Highly customizable to meet specific requirements. 5. Seamlessly integrates with privacy-preserving frameworks like PATE.	1. Lacks built-in differential privacy algorithms. 2. Being relatively new, it has a smaller support community. 3. Does not offer secure aggregation features.
Sherpa.ai	1. Relatively easy to use, thanks to features like Jupyter notebooks. 2. Implements FL algorithms and allows for easy customization.	1. Limited documentation. 2. Small community with only seven contributors. 3. Project repository is not actively updated (last update was over 4 months ago). 4. Restricted to simulation mode. 5. Limited applicability to certain scenarios.
FedML	1. Support for on-device training, including smartphones and IoT devices. 2. Capability for distributed computing. 3. Growing and active community. 4. Support for multi-GPU training.	1. Lacks built-in privacy-preserving techniques, relying solely on secure aggregation. 2. The variety of available modules for different situations could introduce complexity and potential overhead.
PaddleFL	1. Offers a high-level interface for common FL aggregators and includes a differentially private algorithm. 2. Supports various privacy-preserving methods such as Differential Privacy (DP), Multi-Party Computation (MPC), and secure aggregation.	1. Can be challenging to use due to its reliance on a less-known deep learning platform. 2. Features limited documentation and a relatively small community with only 12 contributors. 3. Lacks compatibility with other frameworks, which can be a significant drawback.

LEAF	1. Offers essential Federated Learning mechanisms, including the Federated Averaging Aggregator. 2. Follows a modular and adaptive approach. 3. Supports reproducible science.	1. Lacks benchmarks for privacy preservation in FL. 2. Has limited official documentation and tutorials. 3. Primarily designed for production use rather than research purposes.
NV FLARE	1. Offers Federated Learning mechanisms like the Federated Averaging Aggregator. 2. Adopts a modular and adaptive approach. 3. Facilitates reproducible scientific experiments.	1. User-friendly and includes benchmarks for privacy preservation. 2. Provides official documentation and tutorials. 3. A powerful tool for various Federated Learning tasks.

The comparison among the FL frameworks listed in Table 2 is conducted based on the following criteria:

- *Criterion 1*: This criterion evaluates essential features of Federated Learning frameworks. It considers the framework's compatibility with various operating systems, its support for different Federated Learning scenarios like cross-silo or cross-device setups, the machine learning and deep learning libraries it is compatible with (such as TensorFlow or PyTorch), and whether it includes a Federated attack simulator for assessing security aspects.
- *Criterion 2*: This criterion evaluates the framework's computing paradigms, encompassing three essential aspects. Firstly, it assesses whether the framework supports standalone simulation, enabling users to apply Federated Learning scenarios in a simulated environment. Secondly, it examines the framework's distributed computing capability, determining if it can effectively operate in a distributed environment involving participants using different devices. Lastly, it evaluates the framework's ability to facilitate on-device training, a crucial feature for IoT and mobile devices with typically constrained hardware resources.
- *Criterion 3*: If FL frameworks include standardized FL algorithms and configurations like Federated Average, FedNAS [37] decentralized Fl, vertical Fl, and split learning [38]
- *Criterion 4:* A fundamental attribute for any Federated Learning framework is the inclusion of privacy-preserving mechanisms, and it's equally important to assess the types of privacy-preserving methods supported by these frameworks. In scenarios where a framework lacks built-in privacy-preserving techniques, it should possess the capability to seamlessly integrate such mechanisms.
- *Criterion 5:* For an FL framework to be flexible and adaptive, documentation, tutorials and Community support are significant.
- *Criterion 6:* Secure aggregation algorithm implementation to further enhance privacy.
- *Criterion 7:* In contemporary settings, conducting training processes on GPUs, particularly for Deep Learning tasks, has become indispensable. This is particularly crucial for devices with constrained hardware resources, as GPUs have demonstrated exceptional computational prowess in comparison to CPUs.

- *Criterion 8:* All the FL frameworks in comparison are open-sourced but with different licenses and therefore of different usage limitations.
- *Criterion 9:* Additional overarching attributes and qualities of Federated Learning (FL) frameworks should also be considered. Specifically, an FL framework should prioritize ease of use, adaptability, the preservation of interoperability, flexibility, and privacy to effectively cater to various needs and scenarios.

The characteristics of each FL framework against the 9 identified criteria are tabulated in Table 3 and Table 4.

Table 3: Federated Learning framework comparison (1/2).

FL Framework	TensorFlow Federated	PySyft + PyGrid	FATE	Flower	Sherpa.ai
Standalone simulation	Yes	Yes	Yes	Yes	Yes
Distributed computing	Yes	Yes	Yes	Yes	No
FedAvg	Yes	Yes	Yes	Yes	Yes
Decentralized FL	No	No	No	Yes	No
FedNAS	No	No	No	Yes	No
Vertical Federated Learning	No	No	Yes	Yes	No
Split Learning	No	Yes	No	Yes	No
Privacy-preserving Methods	Differential Privacy	Multi-Party Computation - Homomorphic Encryption	No	Differential Privacy (PATE) - Implemented in NESTLER	Differential Privacy
DP Noise type	No	No	No	Yes	Yes
Adaptive Differential Privacy	No	No	No	No	Yes
Subsampling methods to increase privacy	No	No	No	No	Yes
Documentation and Community support	Yes	Yes	Partial - Mostly in Chinese	Yes, and it's growing rapidly	Yes
Secure Aggregation	No		Yes	Future Implementation	No

Table 4: Federated Learning framework comparison (2/2).

FL Framework	FedML	PaddleFL	LEAF	OpenFL	FLARE
Standalone simulation	Yes	Yes	Yes	Yes	Yes
Distributed computing	Yes	Yes	Yes	Yes	Yes
FedAvg	Yes	Yes	Yes	Yes	Yes
Decentralized FL	Yes	Yes	Yes	Yes	Yes
FedNAS	Yes	No	No	No	Yes
Vertical Federated Learning	Yes	Yes	Yes	Yes	Yes
Split Learning	Yes	Yes	Yes	Yes	Yes
Privacy-preserving Methods	No	Multi-Party Computation	No	Multi-Party Computation & Differential Privacy	Multi-Party Computation & Differential Privacy
DP Noise type	No	Yes	Yes	Yes	Yes
Adaptive Differential Privacy	No	Yes	Yes	Yes	Yes
Subsampling methods to increase privacy	No	Yes	Yes	Yes	Yes
Documentation and Community support	Stable	Partial	Partial	Partial but growing	Partial but growing
Secure Aggregation	Future Implementation	Yes	Yes	Yes	Yes

Based on the tables above, Flower, PySyft & PyGrid, TensorFlow Federated, Sherpa.ai, PaddleFL, and OpenFL offer privacy-preserving methods to varying degrees. However, Sherpa.ai, PaddleFL, and OpenFL lack support for on-device training, which is essential for some NESTLER Living Lab use cases involving edge computing. Taking these factors into account, Flower, PySyft & PyGrid, NV FLARE, and TensorFlow Federated (when enhanced with privacy-preserving methods) appear to be the most suitable options for further analysis and support within the NESTLER project.

3 NESTLER AI Framework (NAIF)

The NESTLER AI framework aims to enable model training through Federated Learning using any Federated Learning (FL) framework. In this context, NESTLER intends to provide a NESTLER AI Framework (NAIF) that allows for the flexible deployment of FL frameworks as needed, providing a seamless and adaptable environment for FL-based model training.

3.1 High Level Architecture

The high-level architecture of NAIF is depicted in Figure 15. As depicted in the figure, the FL API holds a pivotal position in enabling access to the desired FL frameworks. It serves as an interface that external entities, whether they are other NESTLER services, third-party services, or external users, can utilize to access FL-related services. These services encompass functions like setting up and managing an FL framework. Given standardized specifications for describing FL frameworks, this API has the potential to facilitate the deployment of FL node networks and support collaborative training among them.

Without loss of generality, NESTLER AI Framework Backend is currently able to supports three state-of-the-art FL frameworks, namely NVIDIA FLARE [19], Flower [13], which for the sake of privacy preservation has been integrated with PATE [36], and Tensorflow Federated [10]

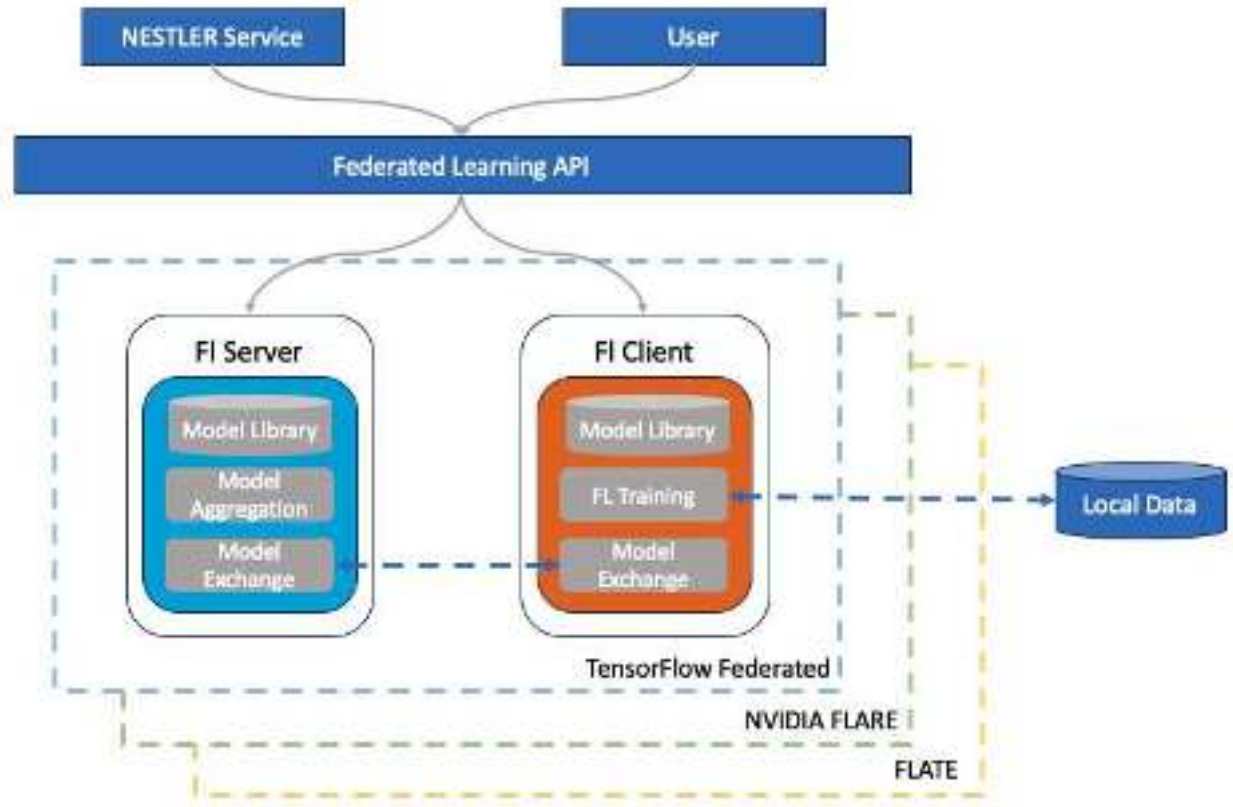


Figure 15: High-level architecture of NAIF

Details about the Federated Learning API, which describes their homogenization under a common API, will be included in next version of the deliverable, namely D4.3 "NESTLER backend implementation of AI algorithms and agricultural services".

In the context of NAIF, a comprehensive comparative analysis of various FL frameworks has been carried out. This analysis involved identifying the key strengths and weaknesses of ten different FL frameworks and evaluating them based on criteria related to their FL features and privacy-preserving capabilities. Based on the analysis, the most prominent candidates for supporting the Living Lab use cases have been identified as:

- ***NVIDIA FLARE***, which was released as open source in late November 2021, represents a significant addition to the world of Federated Learning. It is supported and developed within the NVIDIA community and promises to be a notable FL framework with the backing of a leading technology company.
- ***Flower***, which is a promising candidate for real FL settings and is integrated with PATE by NESTLER to support privacy-preserving FL, named ***FLATE***.
- ***Tensorflow Federated***, which is a well-known and state-of-the-art Federated Learning framework that is suitable for extensive Federated Learning simulations and research tasks. It provides a powerful platform for experimenting with various FL scenarios and algorithms.

In the following subsections, a technical introduction to each of these Federated Learning (FL) frameworks is provided, followed by an overview of the privacy preservation enhancements considered in NESTLER. These introductions aim to help you understand the capabilities and features of each framework, as well as the privacy-preserving mechanisms integrated into NESTLER's use cases.

3.1.1 NVIDIA FLARE

NV FLARE [19] is a FL framework that facilitates Federated Learning with a trusted setup between remote clients. It employs a provisioning tool to help developers establish this setup. The provisioning tool creates startup kits for each site in an encrypted package and sets up communication channels using shared SSL certificates for secure communication between participants. This tool is built on the Open Provision API, and NVIDIA Flare provides various builder modules to streamline the configuration process. The configurations generated by the provisioning tool include essential information to ensure mutual trust and system-wide consistency among participants.

1. Network Discovery: It configures domain names and IP addresses for network communication. Servers are typically identified by fully qualified domain names (FQDNs), but unique hostnames with mapped IPs can also be used.
2. Credentials for Authentication: The provisioning tool handles the creation of certificates for participants, including server, client, and admin certificates. These certificates are then signed by a root Certificate Authority (CA) to establish secure communication channels.
3. Authorization Policy: It defines the roles and rules for participants. This includes specifying the roles of clients (e.g., admin, training-participant) and outlining their respective permissions and restrictions.

4. Tamper-Proof Mechanism: To ensure the integrity of startup kits and configuration files, the tool generates cryptographic signatures. These signatures can be used to verify that no unauthorized tampering has occurred.
5. Convenient Commands: The provisioning tool provides shell scripts with command-line options, making it easy for participants to join the Federated Learning process. Participants can execute these scripts to quickly become part of the FL setup.

Developers have access to the Open Provision API, as shown in Figure 16 which allows them to have full control over the provisioning process. Using this API, developers can customize and manage provisioning tasks to meet their specific requirements. They have the flexibility to add, modify, or remove configurations as needed, ensuring that the provisioning process aligns with their particular use cases and preferences. This level of control empowers developers to tailor the setup of NVIDIA Flare to suit their unique needs and objectives.

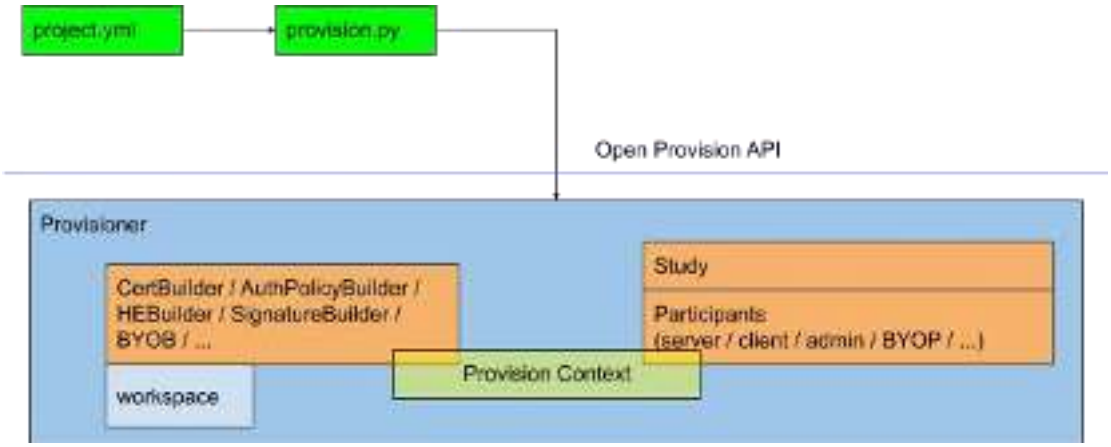


Figure 16: NVIDIA FLARE provisioning Tool Workflow.

Figure 16 outlines the architecture of the Open Provision API. Within the 'project.yml' file, developers have the capability to define participants and builders, as well as store essential information related to the entire implementation process. For instance, developers can specify their preferred API version, identify participants, and designate builder classes. An illustrative example of the 'project.yml' file is provided in Figure 17.

The python file *provision.py* [39] is an application designed to interact with the Open Provision API. The application reads all the relevant data from the 'project.yml' file and creates instances of classes and subclasses as defined by the Open Provision API. Within this application, the Provisioner class serves as a container, housing instances of Project, Workspace, Provision Context, Builders, and Participants.

To provide a concise overview, let's briefly describe the fundamental classes within the Open Provision API. The Project class stores participant information, while the Participant class contains detailed participant data such as names and organizations. The Builder class plays a pivotal role in the entire process and encompasses eight distinct subclasses. Its primary function is to extract information from the project and provisioner, generating commonly used zip files for a standard

NVIDIA FLARE system. These files may include tenseal context files for both the server and client components. For instance, the StaticFileBuilder utilizes data from 'project.yml' to iterate through participants, compile the contents of each file, and replace them with the appropriate values. The HEBuilder specializes in constructing content related to homomorphic encryption.

```yaml
participants:
  # change overseer.example.com to the FQDN of the overseer
  - name: overseer.example.com
    type: overseer
    org: nvidia
    protocol: https
    api_root: /api/v1
    port: 8443
  # change example.com to the FQDN of the server
  - name: example1.com
    type: server
    org: nvidia
    fed_learn_port: 8002
    admin_port: 8003
    # enable byoc loads python codes in app.  Default is false.
    enable_byoc: true
    components:
      <<: *svr_comps
  - name: example2.com
    type: server
    org: nvidia
    fed_learn_port: 8002
    admin_port: 8003
    # enable byoc loads python codes in app.  Default is false.
    enable_byoc: true
    components:
      <<: *svr_comps
  - name: site-1
    type: client
    org: nvidia
    enable_byoc: true
    components:
      <<: *cln_comps
  - name: site-2
    type: client
    org: nvidia
    enable_byoc: false
    components:
      <<: *cln_comps
```

```yaml
# The same methods in all builders are called in their order defined in builders section
builders:
  - path: nvflare.lighter.impl.workspace.WorkspaceBuilder
    args:
      template_file: master_template.yml
  - path: nvflare.lighter.impl.template.TemplateBuilder
  - path: nvflare.lighter.impl.static_file.StaticFileBuilder
    args:
      # config_folder can be set to inform NVIDIA FLARE where to get configuration
      config_folder: config

      overseer_agent:
        path: nvflare.ha.overseer_agent.HttpOverseerAgent
        # if overseer_exists is true, args here are ignored.  Provisioning
        #   tool will fill role, name and other local parameters automatically.
        # if overseer_exists is false, args in this section will be used.
        overseer_exists: true
        # args:
        #   sp_end_point: example1.com:8002:8003

      snapshot_persistor:
        path: nvflare.app_common.state_persistors.storage_state_persistor.StorageStatePers
        args:
          uri_root: /
          storage:
            path: nvflare.app_common.storages.filesystem_storage.FilesystemStorage
            args:
              root_dir: /tmp/nvflare/snapshot-storage
              uri_root: /

  - path: nvflare.lighter.impl.auth_policy.AuthPolicyBuilder
    args:
      orgs:
        nvidia:
          - relaxed
      roles:
        super: super user of system
      groups:
        relaxed:
          desc: org group with relaxed policies
          rules:
            allow_byoc: true
```

Figure 17: Provisioning in NVIDIA FLARE by project.yml

It's important to mention that NV FLARE offers a user-friendly provisioning tool with a graphical interface to streamline interactions with the 'project.yml' file. In this graphical interface, depicted in Figure 18, users can input essential details pertaining to the entire provisioning procedure. This includes specifying the roles and privileges of each participant, their respective names, server IP addresses, and the communication ports they utilize. Once all the necessary information has been provided by the user, they can effortlessly generate the 'project.yml' file.

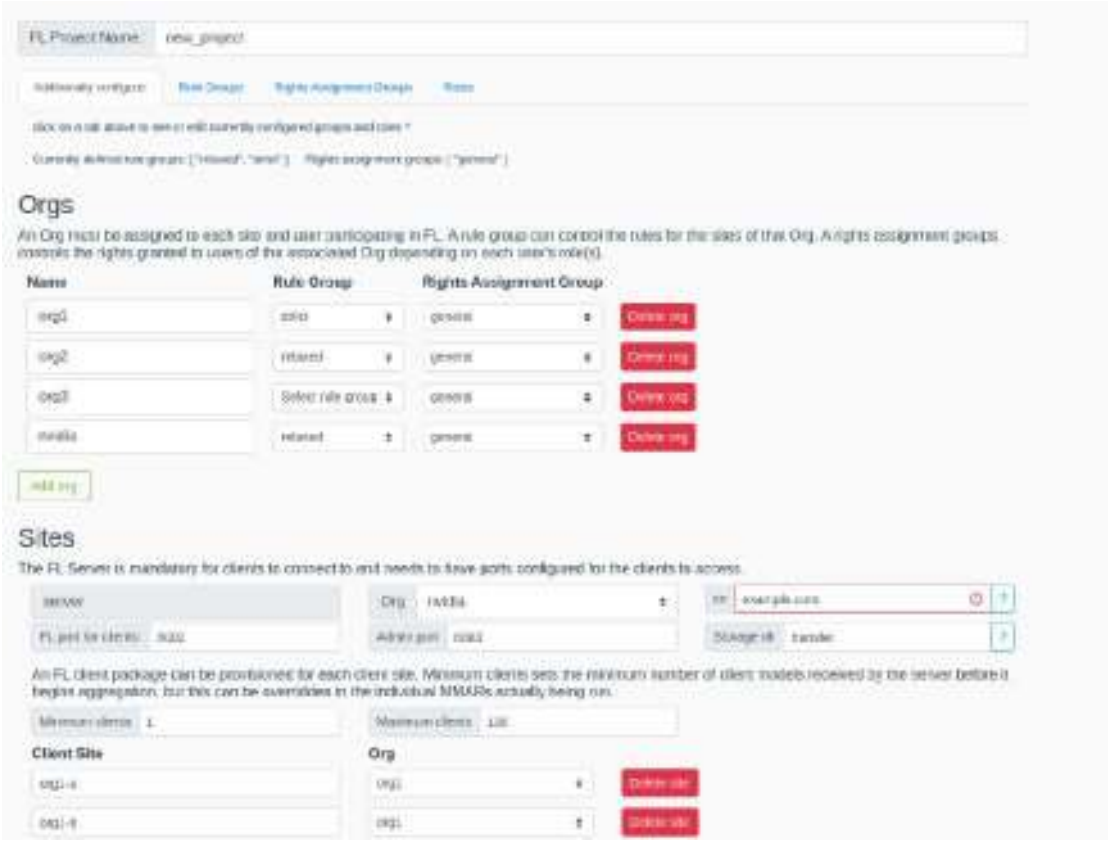


Figure 18: UI of provisioning tool helper in NV FLARE

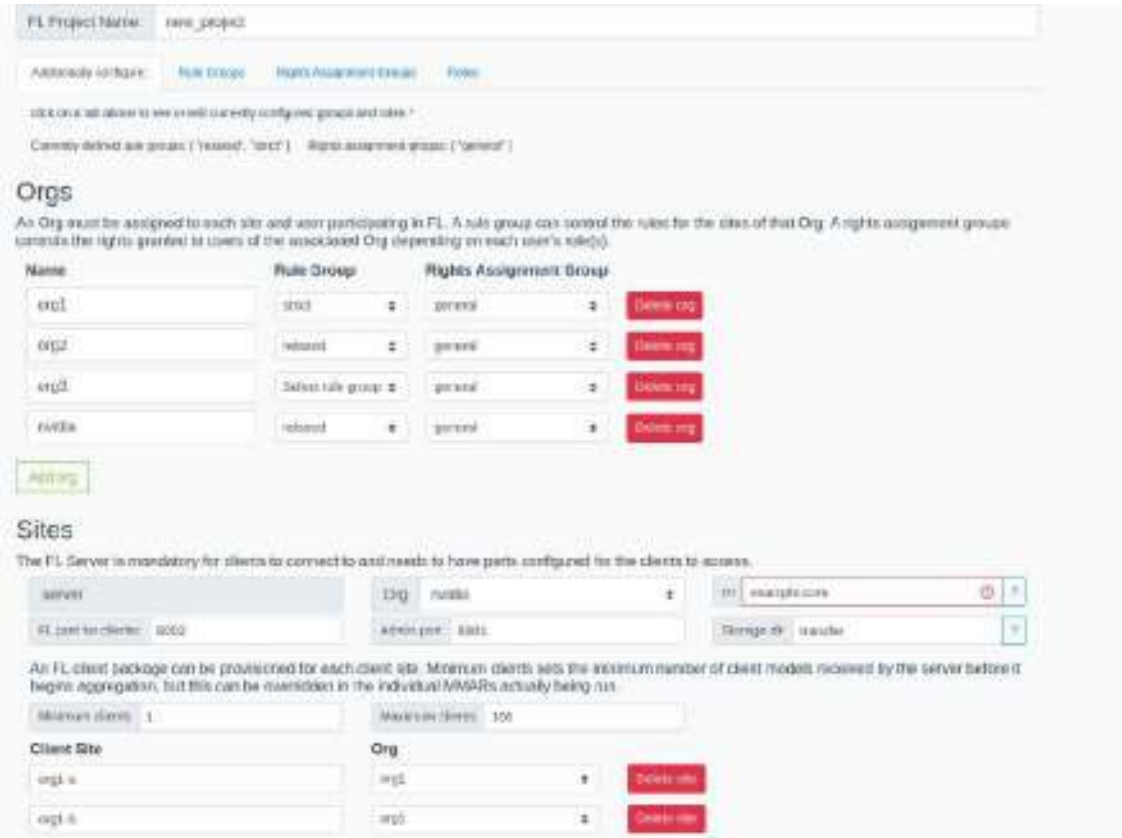


Figure 19: UI of provisioning tool helper in NV FLARE

3.1.2 NESTLER FLATE: Flower & PATE

Flower [40] Flower, an open-source Federated Learning framework, is designed to function across a diverse range of environments, spanning from IoT devices and mobile phones to larger client populations, including thousands of devices. It's important to note that Flower does not come with built-in mechanisms for preserving privacy. In the following sections, we present NESTLER FLATE, an integrated Federated Learning framework that works in conjunction with Flower. FLATE leverages PATE for privacy preservation, but it's essential to recognize that PATE [35] is primarily a transfer learning framework and may not fully address the specific privacy protection requirements inherent in Federated Learning, especially in scenarios involving potentially malicious participants. An alternative framework known as FL-PATE [41] seeks to bridge the gap by combining PATE's transfer learning capabilities with the context of Federated Learning.

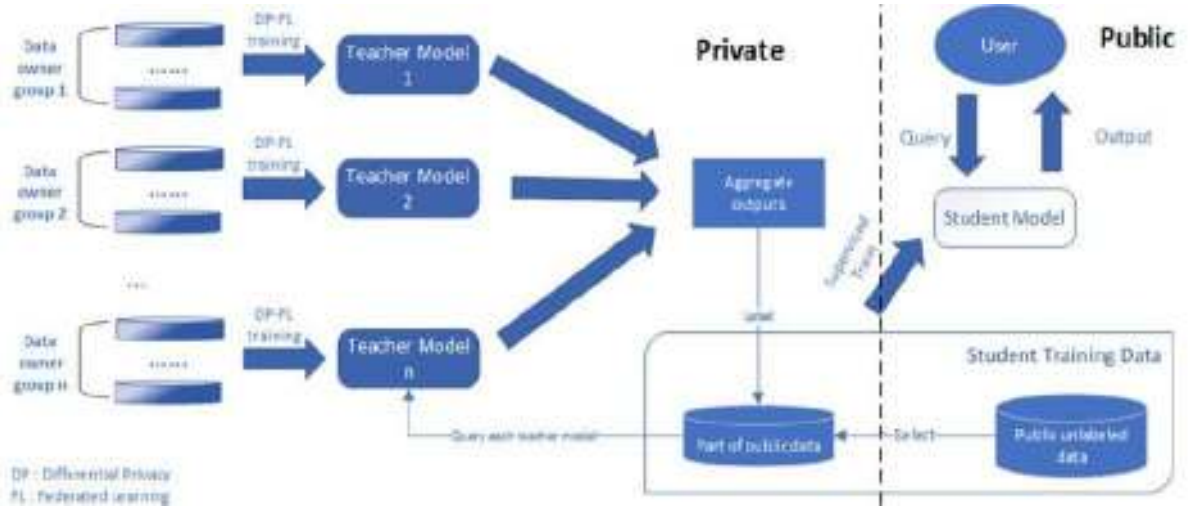


Figure 20: Overview of FL-PATE framework

FL-PATE operates through two distinct stages, as depicted in Figure 20. In the initial stage, Teacher models undergo training using Federated Learning on groups of participants. Differential privacy is introduced by injecting Gaussian noise into a specific layer of the Teacher models, significantly mitigating the impact of the noise on model accuracy. The second stage involves the training of the Student model using publicly available datasets that are labeled with aggregated outputs derived from the Teacher models. It's important to note that FL-PATE leverages PATE as a transfer learning method rather than a privacy-preserving framework. During Federated Learning training, Gaussian noise is applied to the weights of the aggregated model, while PATE introduces Laplacian noise to the predictions generated by the majority-voted teachers. Within the NESTLER framework, we have developed an FL solution that leverages PATE for both transfer learning and privacy preservation. While FL-PATE relies on client-level differential privacy for teacher model training in a federated learning context, PATE employs a vote-level differential privacy mechanism.

FLATE is a Federated Learning framework that builds upon PATE for knowledge transfer and incorporates differential privacy. Similar to PATE, FLATE adopts a teacher-student framework in which teachers are trained within a Federated Learning environment. After the Federated Learning process is completed, each teacher's model performs inference on public data provided by the student. These predictions on public data are aggregated with added noise to train the final Federated Learning framework. In our system, we have a Flower Server and multiple Flower Clients, each of which possesses local private datasets. During the Federated Learning training process, both the teachers and the students function as clients. The Flower server is responsible for training the teacher models and performing noisy aggregation of their outputs to label publicly available unlabeled datasets for the students [42].

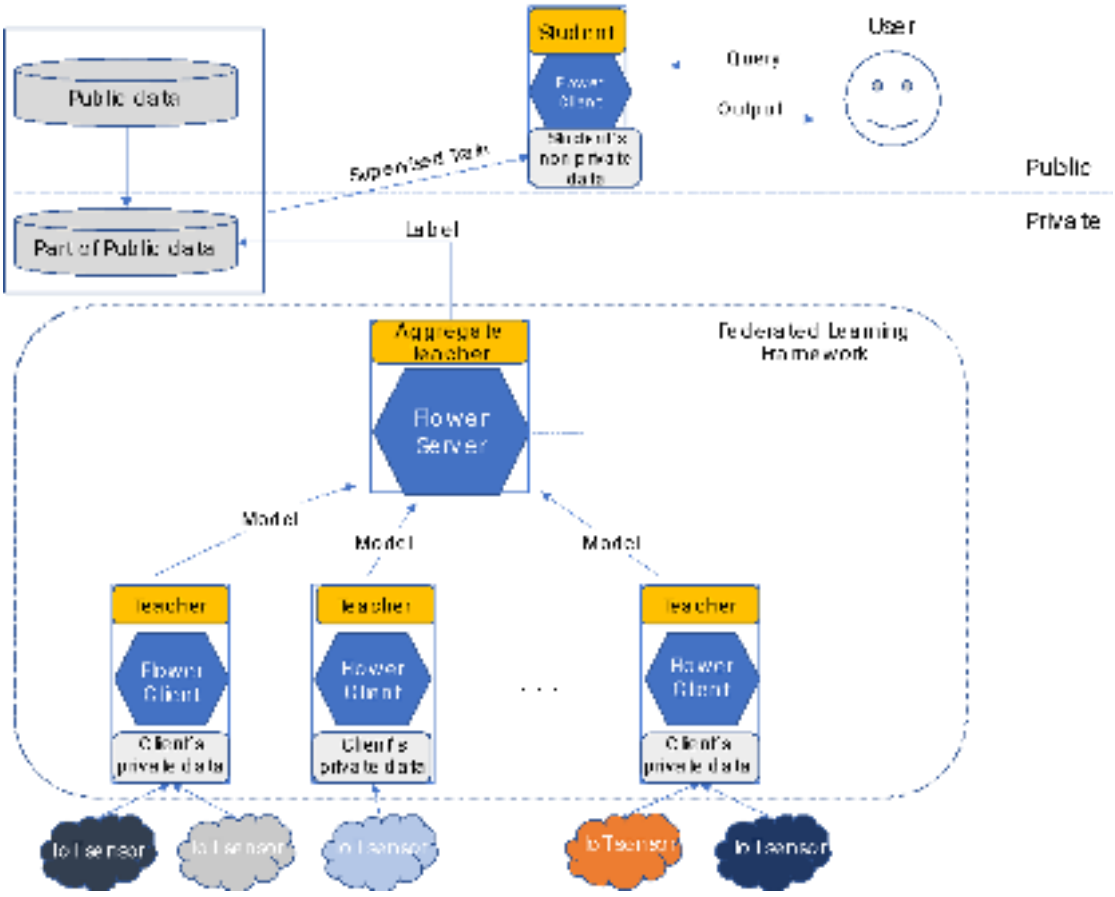


Figure 21: Overview of proposed FLATE framework.

An overview of the design of FLATE is illustrated in Figure 21. The objective is to train a model while preserving privacy, enabling it to handle complex real-world tasks. Each teacher's model is trained using its own sensitive data, which remains confidential. The ultimate outcome of this entire process is the student model, which can be made publicly available. Here's how the process unfolds: The server conducts the training of the teachers within a federated learning framework, while the student awaits the completion of the teachers' training process. During each round of federated learning, the teachers download the global model from the Flower Server and train it using their respective local private-sensitive training datasets, employing the Stochastic Gradient Descent (SGD) training algorithm [43]. Then, the teachers upload the weights $\Delta w \Delta w$ to the *Flower* Server and the global model is updated by the average of $\Delta w \Delta w$. We use $FedAvg$ as the aggregation strategy.

In the final round of Federated Learning (FL) training, the teachers do not upload their models to the server. Instead, these teacher models are used to make predictions on the public data belonging to the student. Subsequently, the server leverages this unlabeled public data from the student to query the teachers and performs noisy aggregation on their respective predictions. This aggregated information is then used to label the student's training set, and the student is trained through supervised learning.

Throughout this entire process, only the student's model is made public, and the teachers' models remain inaccessible to users. The final trained student's model is more resilient to membership inference attacks and model inversion attacks [44] [45]. This is because the ultimate training occurs on public data, and thus, it does not directly expose information about the private data of the teachers. However, it does incorporate knowledge acquired from the private data of each teacher,

as the labels used in the training procedure are derived from the predictions of the teachers' models on the unlabeled data. The student's training dataset is sourced from public, unlabeled data, and the server labels this data through noisy aggregation of the outputs generated by the teacher models on that data. FLATE employs the same noisy aggregation mechanism as PATE.

3.2 Addressing NESTLER Use Cases

Given the wide range of machine learning applications that can be accommodated by the chosen Federated Learning (FL) frameworks, each with its unique performance and privacy demands, a comprehensive understanding of their performance remains elusive for every application. To address this, the three selected frameworks have been thoroughly examined within the context of representative scenarios. These investigations aim to shed light on the utility of the NAIF frameworks, particularly in the context of NESTLER Use Cases (UCs), as outlined in the following Table 5.

Table 5: Summary of AI services and features experimented for the 3 FL frameworks.

FL Frameworks	NVIDIA FLARE	NESTLER FLATE	TensorFlow Federated
AI service	Object detection	Image classification	Tabular data classification
Privacy Preservation technique	Percentile Privacy, Homomorphic Encryption, both	Differential Privacy (PATE)	Differential Privacy
Data heterogeneity	yes	yes	yes

3.2.1 NVIDIA FLARE

The FLARE ecosystem has been employed to train and assess a machine learning-based object detector across various scenarios. In the experiments detailed in this deliverable, the YOLO v5 (You Only Look Once) [46] model serves as the machine learning model for object detection. Specifically, it has been trained to identify objects belonging to the "person" class within the COCO dataset. [47].

3.2.1.1 Experimental Setup

YOLO [48] is one of the most well-known object detection algorithms is YOLO (You Only Look Once), which partitions images into a grid system. Each grid cell is tasked with detecting objects within its boundaries. YOLO is renowned for its combination of speed and accuracy. Over time, multiple versions of YOLO have been developed, with each version demonstrating enhanced performance. The experiments detailed in this deliverable have been carried out using YOLOv5 [46], which is a robust version of the YOLO series released as open-source code.

This version offers several YOLOv5 models, which are compound-scaled variants of the same architecture. Smaller models can achieve real-time frames per second (FPS) on edge devices, while larger models prioritize accuracy and are designed for cloud GPU deployments. Here is a brief description of each variant:

- *YOLOv5n*: The nano model version is the smallest one, and the model's capacity is around 4 MB. It is ideal for mobile solutions.
- *YOLOv5s*: The small model has 7.2 million parameters and is more accurate than the nano one making it ideal for IoT devices.

- *YOLOv5m*: This is the medium-sized model with 21.2 million parameters. A suitable choice for many applications since it provides a good balance between speed and accuracy.
- *YOLOv5l*: The large model of the YOLOv5 family with 46.5 million parameters. It indicates better performance from the previous one.
- *YOLOv5x*: It is the largest model among all with 86.7 million parameters. This model can accomplish the highest MAP (Mean Average Precision) among the 5 but is much slower.

All of the aforementioned models have been trained on the COCO dataset, which contains 80 different classes, for a total of 300 epochs. Additionally, these trained models are readily available and can be utilized as pre-trained models to accelerate the training process.

Initially, our object detection algorithm aimed to detect humans. Consequently, the next task was to identify a suitable dataset. Among the available open-source object detection datasets that include the "person" class, the options included the Pascal Visual Object Class (VOC) dataset and the COCO dataset.

3.2.1.2 PASCAL VOC

PASCAL VOC [49] dataset is a well-known reference dataset in the field of object detection tasks. It comprises approximately 12,000 images that have been annotated with bounding boxes for various objects. Specifically, the dataset includes over 28,000 annotations for 20 basic object classes, including the "Person" class. The initial version of this dataset was introduced in 2005 and featured only four categories. However, the final set of 20 classes was established in 2007, and the last complete version of the dataset was released in 2012.

It is important to note that the PASCAL VOC dataset is designed to be free from systematic biases, such as images with centrally positioned objects, well-illuminated scenes, non-occluded objects, and so on. This ensures that the dataset provides a diverse and realistic representation of object detection challenges.

The PASCAL VOC dataset offers challenging images and high-quality annotations for all object classes. Specifically, for the "Person" class, this dataset includes images that depict individuals engaged in a wide variety of activities. These activities may include walking, riding bikes, sitting on buses, and more. The diversity of activities represented in the dataset ensures that object detection models trained on it can handle a broad range of scenarios involving people. Figure 22 illustrates images with the bounding boxes of objects belonging to the class person.

Figure 22: Annotations of PASCAL VOC dataset

3.2.1.3 COCO

The COCO dataset is a comprehensive and extensive object detection dataset designed to address fundamental challenges in object analysis. It encompasses various core analysis problems, including the detection of objects in non-iconic scenes, contextual reasoning within objects, and precise 2D localization of objects. This dataset comprises a massive collection of 1.5 million object instances spanning 80 object classes. It includes object segmentation data and offers rich information for object-related tasks. Notably, YOLOv5 has already been trained on the COCO dataset, achieving satisfactory performance in object detection.

The annotation process for individuals in the "person" class is carried out meticulously. Each image is annotated with bounding boxes for each individual person present, even in scenarios with multiple people.Figure 23 provides visual examples of these annotations. The left image illustrates the annotation for a single person, while the right one demonstrates the annotation for a group of people.

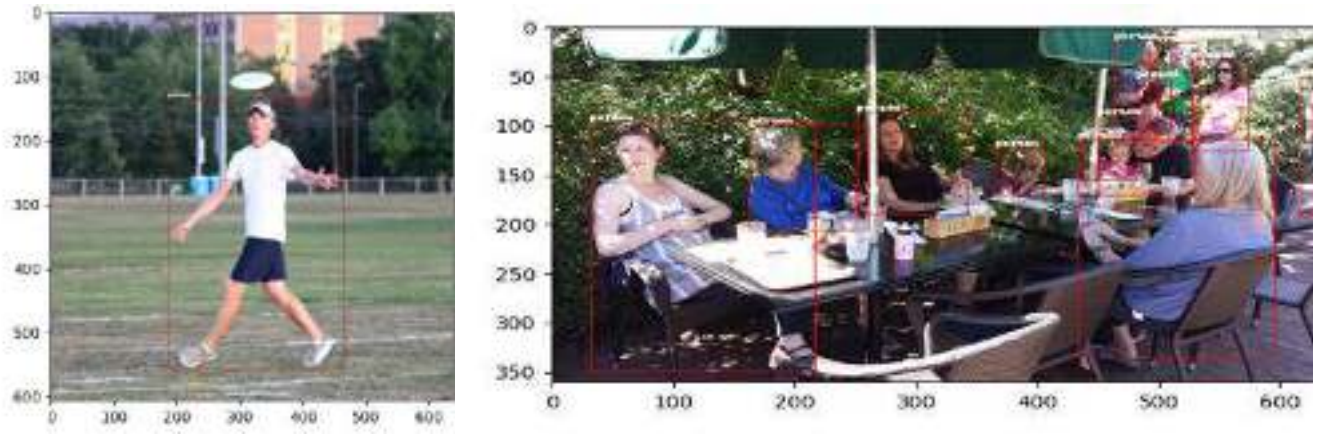


Figure 23: Annotations of the MS Coco dataset.

Within the context of NESTLER, the YOLOv5s model has been chosen from the YOLOv5 series due to its compact architecture, which makes it suitable for training on IoT devices. This model is initially based on a pre-trained model using the COCO dataset. Subsequently, it undergoes Federated Learning (FL) training and evaluation using relevant test and evaluation datasets derived from the VOC dataset. To accomplish this, VOC images containing people need to be identified, and their associated files should be converted into the appropriate format.

The most commonly used metric for assessing the performance of object detection models like YOLOv5 is the mean Average Precision (mAP). To calculate mAP, Intersection over Union (IoU) and the Precision-Recall curve are combined. Initially, IoU is employed to measure the overlap between predicted bounding boxes and ground truth, leading to the calculation of Precision and Recall. Subsequently, a Precision-Recall curve is plotted, and the Average Precision (AP) is determined by computing the area under the curve through segmentation of recalls. The mAP is then derived as the average of AP values calculated for all classes and is defined as follows:

$$MAP = \frac{\sum_q^Q AveP(q)}{Q} \quad (1)$$

Where Q is the number of queries in the set and the AveP(q) is the average precision (AP) for a given query, q.

The training of the YOLOv5s model is carried out using the NVIDIA FLARE framework. Each client involved in the training process has its own dataset from the VOC dataset. To improve training efficiency, pre-trained weights provided by YOLOv5s are utilized. The NVIDIA Flare framework offers two modes for ML model training: the Proof of Concept (POC) mode and the Secure FL workspace mode using the provisioning tool. Initially, a portion of the experiments is conducted in the POC mode. POC serves as a FL Simulator and is accompanied by a set of tools for deploying and managing production workflows. This mode allows developers to locally test the Federated Learning process and assess its performance. All experiments involve a total of 3 clients to evaluate their response to the FL process. Additionally, a part of the experiment is implemented using Provisioning.

3.2.1.4 Installation Guidelines

This section contains the necessary steps to run the integration of Yolov5, with NV FLARE.

Installation

```
1.  cd nvidia-flare-yolov5-integration
2.  conda create --name <flare-yolo> python=3.8
```

```
3. conda activate <flare-yolo>
4. pip install -r requirements.txt
```

Change basic parameters

1. Edit the NVFlare/examples/nvidia-flare-yolov5-integration/yolov5/custom/data/myfilename.yaml file by adding your dataset directory path, number, and names of training classes.

```
path: home/user/datasets/mydataset  # dataset root dir

train: images/train  # train images (relative to 'path') 128 images

val: images/val  # val images (relative to 'path') 128 images

test:  images/test

# Classes

nc: 2  # number of classes

names: [ 'person', 'car']  # class names`
```

2. Go back to the *NVFlare/examples/nvidia-flare-yolov5-integration/yolov5/config/* and edit the *config_fed_client.json* and *config_fed_server.json* files. On the client's *json* file you can insert:

   - **"imgsz"**: image size of your data
   - **"epochs"**: number of epochs to train for every round
   - **"batch_size"**: batch size
   - **"num_clients"**: the number of clients
   - **"pretrained":** use pretrained weights (Yes or No)

   On the server's json file you can insert:

   - **"min_clients"**: minimum number of clients
   - **"num_round"**: the number of rounds

   Also, on the server's json file you can choose if you want a pretrained model with pretrained weights or a model with randomized weights (begin the whole training process from scratch). In the json file insert:

   - *For pretrained network*:

   "path": *yolonetwork.YoloPretrainedNetwork*

   - *For no-pretrained network***:**

   "path": *yolonetwork.YoloNetwork*

How to run

Go back to the root folder and begin the federating learning process in POC (Proof of Concept) mode. The example below is for 2 clients.

```
1. poc -n 2
2. mkdir -p poc/admin/transfer
```

```
3.  cp -rf NVFlare/examples/* poc/admin/transfer
```

Once you are ready to start the FL system, you can run the following commands to start the server and client systems.

- The first step is starting the FL server:
  `./poc/server/startup/start.sh`
- Once the server is running, open a new terminal and start the first client:
  `./poc/site-1/startup/start.sh`
- Open another terminal and start the second client:
  `./poc/site-2/startup/start.sh`
- In one last terminal, start the admin client:
  `./poc/admin/startup/fl_admin.sh localhost`

With the admin client that started from *fl_admin.sh* file, you can control the whole process. You can check the status of the clients and server, you can start the process, stop the process etc. With the admin client command prompt successfully connected and logged in automatically, enter the command.

```
1.  submit_job nvidia_flare-yolov5-integration/yolov5
```

In case you want to abort the process enter the command.

```
1.  abort_job job_id
```

Where job_id is the id that the admin gave you when you started the process. When the whole process ends enter the commands below to shut down the clients and the server. The password is *admin*.

```
1.  shutdown client
2.  shutdown server
```

3.2.2 NESTLER FLATE

In this section, FLATE is used for image classification. The CIFAR-10 dataset [50] used for experiment is a set of images that are commonly used to train machine learning and computer vision algorithms. It is one of the most well-known datasets for machine learning research.

3.2.2.1 Experimental Setup

The CIFAR-10 dataset comprises a total of 60,000 color (RGB) images, each measuring 32x32 pixels, and is divided into 10 distinct classes. Table 6 provides information on the dataset's overall size, as well as the sizes of the training and testing sets. The dataset includes images belonging to the following 10 classes: airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. Each class contains 6,000 images.

Table 6: Number of images in the CIFAR-10 dataset

Dataset	Total Images	Training set size	Test set size
CIFAR-10	60000	50000	10000

In order to train a machine learning (ML) model within a Federated Learning setup involving multiple clients, it's necessary to partition the CIFAR-10 training dataset into separate, non-overlapping sets. Each of these sets is then used to train a Teacher model independently. The number of Teachers in this setup is denoted as "T." The CIFAR-10 test dataset, on the other hand, serves as a source of public, unlabeled data for the training of the Student model. A portion of the labels for this dataset is obtained from the most voted predictions made by the Teachers on this data. To be more specific, 90% of the CIFAR-10 test dataset is utilized in the training process for the Student model, while the remaining 10% is reserved for evaluating the overall training procedure. This approach is provided in Table 7.

Table 7: CIFAR10 dataset partitions for student's training

Dataset	Total Images	Teachers' training set size	Student's training set size	Student's test set size
CIFAR-10	60000	50000	9000	1000

The Deep Learning model trained in FLATE is a Convolutional Neural Network (CNN) with a specific architecture. It comprises three convolutional layers, each consisting of two 3x3 layers, one Max-Pooling layer, and one Batch Normalization layer. Following these convolutional layers, there are three fully connected layers. To evaluate the performance of this machine learning model, the accuracy metric is used. Accuracy is calculated by dividing the number of correct predictions by the total number of predictions made by the model. This metric provides insight into how well the model is performing in terms of making correct classifications.

3.2.2.2 Experimental results

The baseline model is trained in a Federated Learning scenario without the use of any privacy-preserving mechanisms. This training is conducted with only one client (centralized case) for a total of 20 epochs. The training and test datasets used are the same as those employed with the student model. The baseline model achieves an accuracy of 79%. This performance serves as the upper bound for evaluating the Federated Learning algorithms and acts as a baseline for subsequent experiments. Figure 24 shows the accuracy of the baseline model for each epoch, which is trained in a centralized manner.

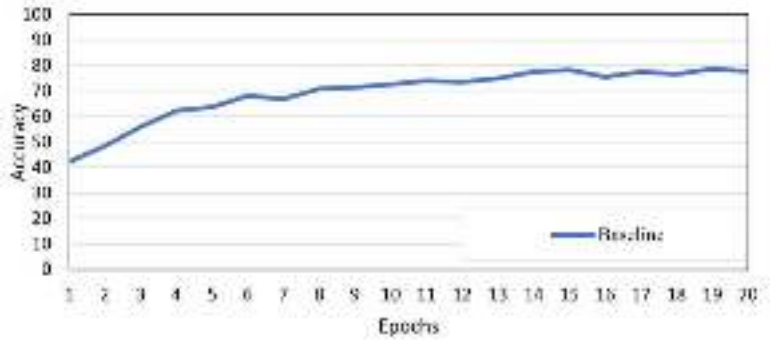


Figure 24: Performance of the student baseline model of FLATE.

We have formed two ensembles, one with 10 teachers and the other with 14 teachers. Accordingly, we have partitioned the dataset into 10 and 14 equally-sized, separate subsets, each containing 5000 and 3571 images, respectively. Each teacher undergoes training for 10 federated rounds, with each round consisting of 10 epochs. Figure 25 provides a visual representation of the test accuracy of the aggregated model after each federated round during the training of the teachers in both ensembles.

In the final round, it achieves an accuracy of 89.5% for the ensemble with 10 clients and 89.2% for the ensemble with 14 clients.

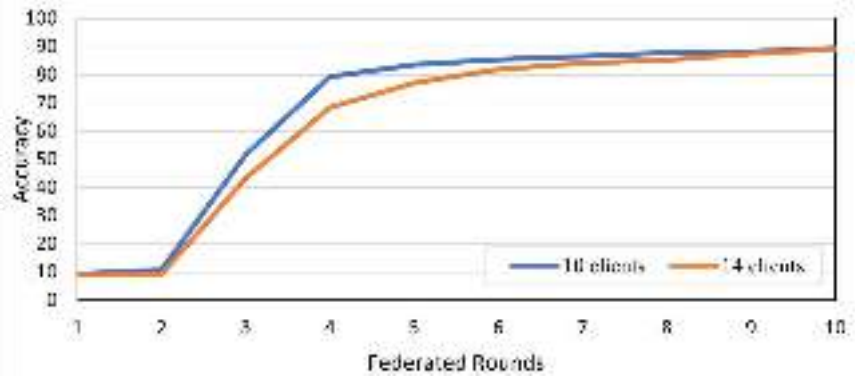


Figure 25: Performance of the aggregated model for each federated round.

3.2.2.3 Installation Guidelines

There are two ways to run the FLATE:

1. With the Conda environment, version >= 4.11.0.
2. With Docker, version >= 19.03.8.

A description of both ways is given in the following.

Conda Environments

a) First, install the project's dependencies
b) Install PyTorch from https://pytorch.org/ depending on your system's requirements.
c) Run the server of the FL system:

```
1. python3 src/py/server.py --net_name=test_net --num_rounds=10 --num_teachers=14 --epochs=10 --
   epsilon=0.2 --num_queries=9000 --noise_data=9000
```

d) Open as many terminals as the number of teachers + 1 are (Teachers + Student),and run the teachers and the student with the command:

```
1. python3 src/py/PATE_pytorch_client_2.py --teacher_id=0 --net_name=test_net --num_rounds=10 --
   num_teachers=14 --teacher_epochs=10 --student_epochs=20 --epsilon=0.2 --num_queries=9000 --
   noise_data=9000
```

where:

- `--net_name`: str, name of your model.
- `--num_rounds`: int, number of rounds for FL.
- `--num_teachers`: int, the number of teachers/clients.
- `--noise`: str, type of noise for the aggregation mechanism ('laplacian' or 'gaussian', default='laplacian').
- `--teacher_epochs`: int, number of epochs to train the teachers.
- `--student_epochs`: int, number of epochs to train the student.
- `--epsilon`: float, epsilon for laplacian distribution (if `--noise='laplacian'`).
- `--sigma`: float, standard deviation for gaussian-normal distribution (if `--noise='gaussian'`).

- `--num_queries`: int, number of queries to the student.
- `--noise_data`: int, the number of data to add noise (must be less or equal than num_queries).`--teacher_id`: int, the ID of the current teacher (must be in range 0 - num_teachers, where teacher_id = num_teachers is the Student)

Docker

a) Set up the docker container.

```
1. docker run -p 8888:8888 -v ~/your_path_to_flower-pate-integration/src/py:/flower -it
   synelixis/flower-pate:v1.8
```

b) Open your browser and go to the *127.0.0.1:8888* address and enter the token that the terminal gave you. You should see a *JupyterLab* running.

c) Run the flwr server of the FL system by opening a new terminal inside the JupyterLab and running:

```
1. python3 /flower/server.py --net_name=test_net --num_rounds=10 --num_teachers=14 --epochs=10 --
   epsilon=0.2 --num_queries=9000 --noise_data=9000
```

d) Open as many terminals as the number of teachers + 1 are (Teachers + Student), inside the JupyterLab and run the teachers and the student with the command:

```
1. python3 /flower/PATE_pytorch_client_2.py --teacher_id=0 --net_name=test_net --num_rounds=10 --
   num_teachers=14 --teacher_epochs=10 --student_epochs=20 --epsilon=0.2 --num_queries=9000 --
   noise_data=9000
```

3.2.3 TensorFlow Federated

In this section, we detail the experiments conducted in federated learning scenarios using TensorFlow Federated. Specifically, we have implemented a Network Intrusion Detection System (IDS) that is trained and deployed in a federated manner. To facilitate training and testing, we utilized the NSL-KDD dataset [51] which contains packet logs encompassing six distinct types of attacks as well as normal records. The results of our experiments demonstrate the robustness of this approach, as it is capable of detecting malicious packets in advance, thereby enhancing cybersecurity in IoT systems.

3.2.3.1 Experimental setup

The NSL-KDD dataset originally comprises 257,673 cyber-attacks categorized into five different classes: DoS, Probe, U2R, R2L (representing attack types), and Normal (indicating no attack). To adapt the dataset for our purposes, we transformed it into a binary classification problem with two classes: "DoS" (Denial of Service attacks) and "No DoS," which encompasses normal traffic and other types of attacks. Our proposed model architecture adheres to the standard Federated Learning training process using the TFF (TensorFlow Federated) framework. Initially, a central server loads the initial global model, which serves as a generic starting point. In each federated round, the server dispatches the model to its clients, and each client proceeds to train the model using its local, sensitive dataset. Upon completing the training process, each client adds Gaussian noise to the model's weights as part of the Local Differential Privacy mechanism. Subsequently, the clients transmit the noisy-aggregated model updates back to the central server. The server then updates the global model by averaging these updated weights using the FedAvg method. After the model's training process is complete, it can effectively classify whether a "DoS" attack is present or not. This model can be applied to IoT systems to detect anomaly attacks in network traffic, as demonstrated in Figure 26.

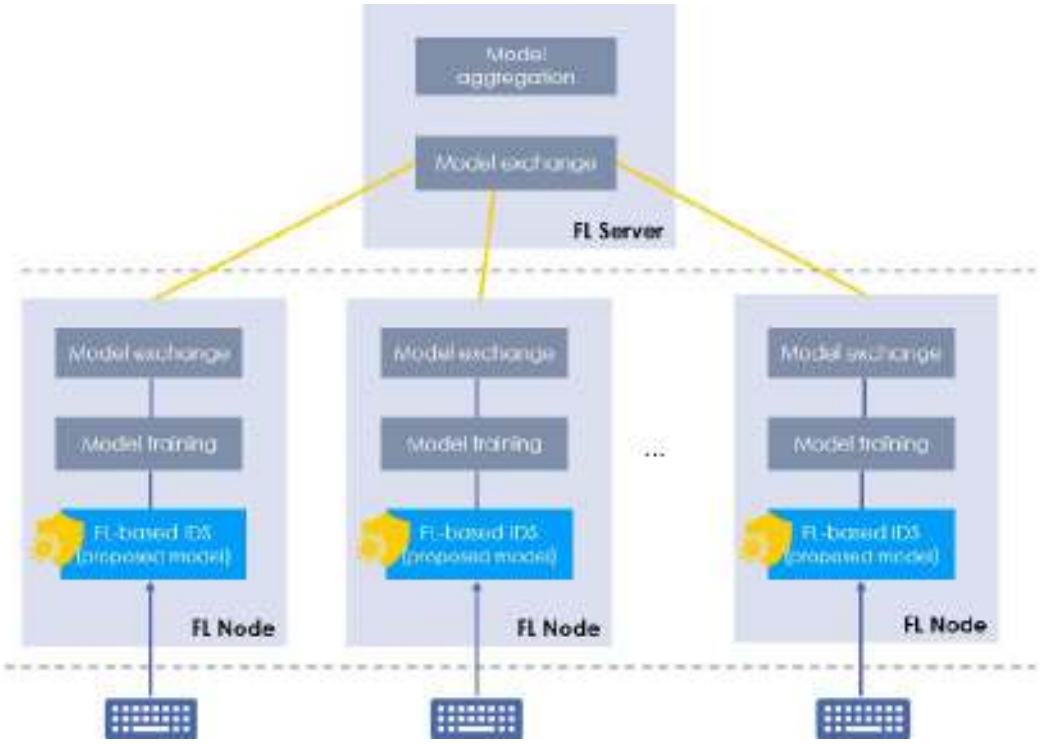


Figure 26: Proposed architecture of FL in an intrusion detection system.

3.2.3.2 Experimental Results

In our experimentation, we conducted a range of experiments to assess the robustness of the Federated Learning intrusion detection model across various real-world scenarios. We focused on testing specific parameters, including the imbalance ratio, the number of clients involved, and the application of differential privacy. Initially, these scenarios were tested independently to understand their individual impacts, and subsequently, we explored combinations of these scenarios to model more realistic conditions. As an illustrative example, we considered different values for the number of clients in our experiments, specifically 1 (representing the baseline model), 10, 25, and 50. These experiments helped us gain insights into how varying the number of clients affects the model's performance and robustness in the context of intrusion detection.

Figure 27 demonstrates the influence of the number of clients on the accuracy of the intrusion detection model. In the testing phase, we achieved the following total accuracies for various numbers of clients: 79.06% for 1 client (baseline model), 78.95% for 10 clients, 77.88% for 25 clients, and 75.54% for 50 clients. It is evident from the results that a higher number of clients leads to a reduction in accuracy. This outcome is expected because dividing the dataset into smaller subsets diminishes certain data relationships that are essential for accurate classification. The difference in accuracy can be substantial, with variations as high as 3.52%.

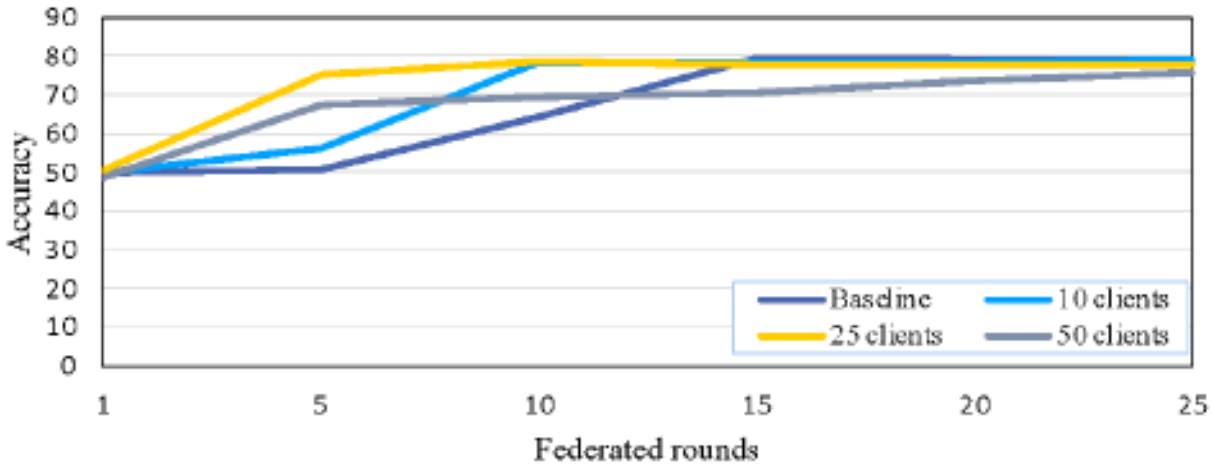


Figure 27: Testing accuracy for different numbers of clients.

3.2.3.3 Installation Guidelines

Set-up

To correctly set up this project, one needs to have the right environment for library versions, python version, etc. To achieve this the following steps must be followed:

- Create a virtual environment
- Install requirements from requirements.txt
- Have python 3.8.10

In our case we used conda environment and pip as the package manager, so an example of this procedure is (after navigating to the project's folder):

- conda create -n venv python==3.8.10
- conda activate venv
- pip install -r requirements.txt

Model Running

To run the code, one needs to run the following command which enables argument parsing. The arguments customize the learning process and allow multiple and different experiments to be conducted by the user:

```
1.  python3 src/tff.py --train_path <train_path> --test_path <test_path> --imb_parameter
    <imb_parameter> --dp_parameter <dp_parameter> --n_clients <n_clients>
```

where:

- `--train_path`: path to train dataset, e.g. /path/to/training-set.csv
- `--test_path`: path to train dataset, e.g. /path/to/testing-set.csv
- `--imb_parameter`: parameter that defines class imbalance. A value of one (1) is perceived as no imbalance, while a value higher than one (1) means higher data imbalance distribution.
- `--dp_parameter`: parameter that defines application of differential privacy. Zero (0) stands for no DP, while a value higher than zero (0) means stronger application of DP.

`--n_clients`: the number of FL clients

3.3 Discussion

Based on the insights gained from the practical experiments with the three Federated Learning (FL) frameworks, as well as the analysis and comparison of existing FL frameworks through desk research, Table 8 summarizes the key lessons learned and highlights the main advantages of each framework. These lessons and benefits are derived from the hands-on experience and provide valuable insights into the strengths of each FL framework.

Table 8: Lessons learnt from FL frameworks' experimental assessment.

FL Framework	Main Benefits
NVIDIA FLARE	• Suitable for real-world scenarios with a single server and multiple remote clients, maintaining data privacy. • Offers various privacy-preserving methods like Percentile Privacy, Homomorphic Encryption, and MPC, which can be used individually or in combination. • Customizable and compatible with popular ML frameworks like TensorFlow and PyTorch for flexible model integration.
NESTLER FLATE	• Suitable for real-world scenarios with a single server and multiple remote clients (Teachers and Student). • Supports scenarios where the Student's data can be partially or completely unlabeled. • Offers enhanced privacy as the final trained model (Student's model) is not directly exposed to sensitive client data. • Provides resistance against membership inference attacks and model inversion attacks, enhancing overall security. • Directly integrates ML algorithms implemented with PyTorch.
TensorFlow Federated	• Mainly suitable for research purposes, allowing ML researchers to conduct experiments and assess FL process performance across different scenarios. • Supports the implementation of Differential Privacy for privacy preservation. • Allows direct integration of ML models developed using TensorFlow.

Next, Table 9 discusses the potential applications of Federated Learning (FL) in NESTLER use cases and offers recommendations for utilizing the three analyzed FL frameworks in these scenarios. The table is not exhaustive but serves as a tool to identify real-world FL applications.

Table 9: Recommended uses of FL in the LL use cases.

Use Case	NVIDIA FLARE	NESTLER FLATE	TensorFlow Federated
Crop diseases prediction & irrigation precision	**train the crop disease predictions, using private smart farm data**	train the crop disease predictions, using private smart farm data and use them to train over public datasets e.g., collected/shared at district/national/ regional level	research on crop disease prediction across hundreds of nodes; research on the effect of model and data poisoning attacks on FL system based

			on GAN-based data generator

4 NESTLER Integration framework and tools

This section focuses on the definition and description of a structured, yet flexible integration framework, able to cater for all NESTLER's subcomponents. Beginning with an introduction to the NESTLER DevSecOps approach, i.e. development, security, and operations, we proceed to an initial description of NESTLER's integration framework that serves as its enabler, consisting of various utilities and workflows that facilitate essentials for the realization of every comprising component, i.e.:

- Source Code Management
- Continuous Integration and Continuous Deployment
- Issue Tracking
- Containerization

In addition to these, a quick overview of the utilized deployment platform is given. In the context of the present document, we focus on the initial approach to DevSecOps and the NESTLER integration framework. We plan to approach the integration framework as a constantly adapting and evolving part of NESTLER throughout the entirety of the project's duration, with updates, in-depth descriptions, and documentations to follow as needed, according to the requirements of the NESTLER components to be realized and integrated, thus forming NESTLER platform as-a-whole.

4.1 NESTLER DevSecOps Approach

DevSecOps, i.e. development, security, and operations, integrates security at every phase of the software development lifecycle, from the initial design to integration, testing, deployment, and software delivery.

DevSecOps transforms security into an innate part of the Agile and DevOps practices adopted by development organizations. In this manner, security issues are handled as soon as they are detected, instead of relying on Quality Assurance (QA) testing at the end of every development cycle, in the production environment. Therefore, through DevSecOps, software development, as well as release, cycles are accelerated by automating the delivery of secure software without slowing the software development cycle.

According to IBM, some of the best practices in DevSecOps include [52]:

- **Shift left.** This term refers to the movement of security from the right (end) to the left (beginning) of the software delivery process. In a DevSecOps environment, security is an innate part of the development process from the beginning. The cybersecurity architects and engineers become part of the development team, ensuring every component and configuration is configured securely.
- **Security education.** Security is a combination of engineering and compliance. Everyone involved with the delivery process of a software component should be familiar with the fundamental principles of application security, application security testing and other security engineering practices.
- **Culture: Communication, people, processes, technology.** The importance of security of processes should be communicated in a concise manner to all participants of a software development team, from product owners to engineers and developers.
- **Traceability, auditability, and visibility.** Traceability allows for tracking of configuration items across the development cycle. Auditability is essential for ensuring compliance with security controls. Finally, visibility is generally considered a good management practice, and is

extremely important in a DevSecOps environment from the aspect of solid monitoring system to measure the heartbeat of the operations.

Given the best practices, the stages of the NESTLER DevSecOps approach include:

- **Plan and create.** Processes are directly related to the software design and development procedures.
- **Verify, package, and release.** These stages are directly related to the Continuous Integration and Continuous Delivery, covered by automated CI/CD tools, and specifically GitLab.
- **Configure, detect, respond, predict, and adapt.** These stages mostly refer to QA testing on production, as well as the processes related to communicating them to the design and development phases again, in a continuous, circular interaction.

During the DevOps steps, monitoring and analytics tools will allow for proper logging in the whole software lifecycle. Last, but not least, Security is introduced within every process of this DevOps cycle, implementing security by design among people, processes and technologies involved in the NESTLER software development and release.

4.2 Source Code Management

Source code management (SCM) refers to tracking modifications on the source code, possibly applied in parallel over a common code base. Besides the obvious benefits of tracking, SCM is of vital significance when resolving conflicts in parallel development streams. The easy access to code, snippets or changes on different code versions facilitates the collaboration among developers, typically working remotely. The main and fundamental characteristics (offerings) of SCM are summarized in the following:

- Complete long-term change history of every tracked file
- Branching and merging features
- Traceability

Source Code Management in the context of NESTLER will be based on Git, given that its structure is aligned with the project needs for parallel development in multiple branches, and code integration. Indeed, in the context of NESTLER, a self-hosted GitLab instance has been deployed to cater for the essential features of SCM, as well as issue tracking and CI/CD, which will be described in the following. NESTLER's GitLab instance is hosted at:

https://git.nestler-project.eu.

In **Error! Reference source not found.**, an overview of NESTLER's GitLab groups and projects can be found.

Figure 28: Overview of NESTLER group in NESTLER's self-hosted GitLab instance

4.3 Continuous Integration and Continuous Deployment

Continuous Integration (CI) is the practice of code integration into a shared repository and frequent automated building / testing of each alteration. Alternatively, CI implements continuous processes of applying quality control, including but not limited to, a set of software development practices, behaviours, and principles for automation and improvement of integration, and certify software continuously.

CI is usually directly linked to the underlying Source Code Management (SCM) system or distributed version control system, i.e. Git in case of NESTLER. In this context, distribution is a facilitator for collaboration, by allowing developers to work on different branches of code or on the master (or main) branch. Code alterations can be committed without affecting the main repository. Hence, developers are allowed to test a pipeline with their own contributions, and without necessarily incorporating someone else's code. Furthermore, distribution allows for parallel development of features, new functionalities, or bug resolutions, as well as discrete and concurrent release cycles for development and testing. GitLab, i.e. the platform of choice for NESTLER, offers added value services on top of Git.

Continuous Delivery refers to release management practice in which software is built in a manner that allows for the production releases at any time. Continuous Delivery is accomplished by continuously integrating late software, building executables, if / when needed, and running automated tests on those executables. These executables are pushed in environments that imitate production ones, to ensure the software will work in a production environment as well. Of course, these production-like environments might imitate the architecture, or deployment platform of the real production environment but not the entirety of its computational resources.

Continuous Deployment is an additional step, providing automated deployment of the application, without human intervention whatsoever. The term Continuous Delivery and Deployment (CD) encapsulates the automated execution of both stages. GitLab offers CD capabilities via the integrated GitLab-CI service [53].

Since both CI and CD practices are highly interconnected, CI/CD refers to integrating the procedures related to CI and CD into a single multi-stage pipeline. In the context of GitLab, the essence of continuous integration, delivery and deployment are the *pipelines* that describe sequential CI and CD operations. Pipelines are composed of *jobs* and *stages* that describe the sequence of *job* execution.

The GitLab *jobs* are executed by GitLab *runners*, while *jobs* of the same *stage* can be executed concurrently, assuming the existence of multiple *runners*. If all *jobs* of a *stage* are successful, then the pipeline moves to the next *stage*; otherwise, it is terminated early, if even one *job* fails. The *jobs* are executed via GitLab *runners*, which run the code defined in YAML scripts with a file named "`.gitlab-ci.yml`" at the root of a repository.

A common development workflow within GitLab is presented in Figure 29 [54]. The code developments made in a local branch are committed and pushed in a branch of a remote repository in GitLab, an action which triggers the CI/CD pipeline set for the specific project:

- The developer implements a certain piece of software to be integrated into the core system functionality.
- The developer then introduces one or multiple test cases that unit-test the new piece of code at the level of rationality checking and consistency of outcomes. The unit-testing should be as thorough as possible.
- The developer commits and pushes code developments on their local Git repository to a branch of the remote GitLab repository.
- The developer creates a merge request for their piece of code.
- The CI/CD pipeline set for the specific project is triggered. The CI platform performs the determined unit-testing.
- If the unit-testing is successful, the code is subject to further integrated system testing.
- If the integrated system testing is successful, then the merge request is accepted and the newly committed source code becomes part of the repository's master (main) branch.
- If the result of the testing is unsuccessful, the merge request is rejected.
- If either the unit or integrated system testing fails, the merge request is rejected. In this case, the developer should consider either a code rewrite to satisfy the unit-testing procedures, or the unit-test rewrite, in case of erroneous testing. The workflow is re-initiated.
- The source code management platform moves to the CD part, building the package and deploying it.
- After successful CD, the new code is running on the deployment infrastructure and is subject to user testing/production.

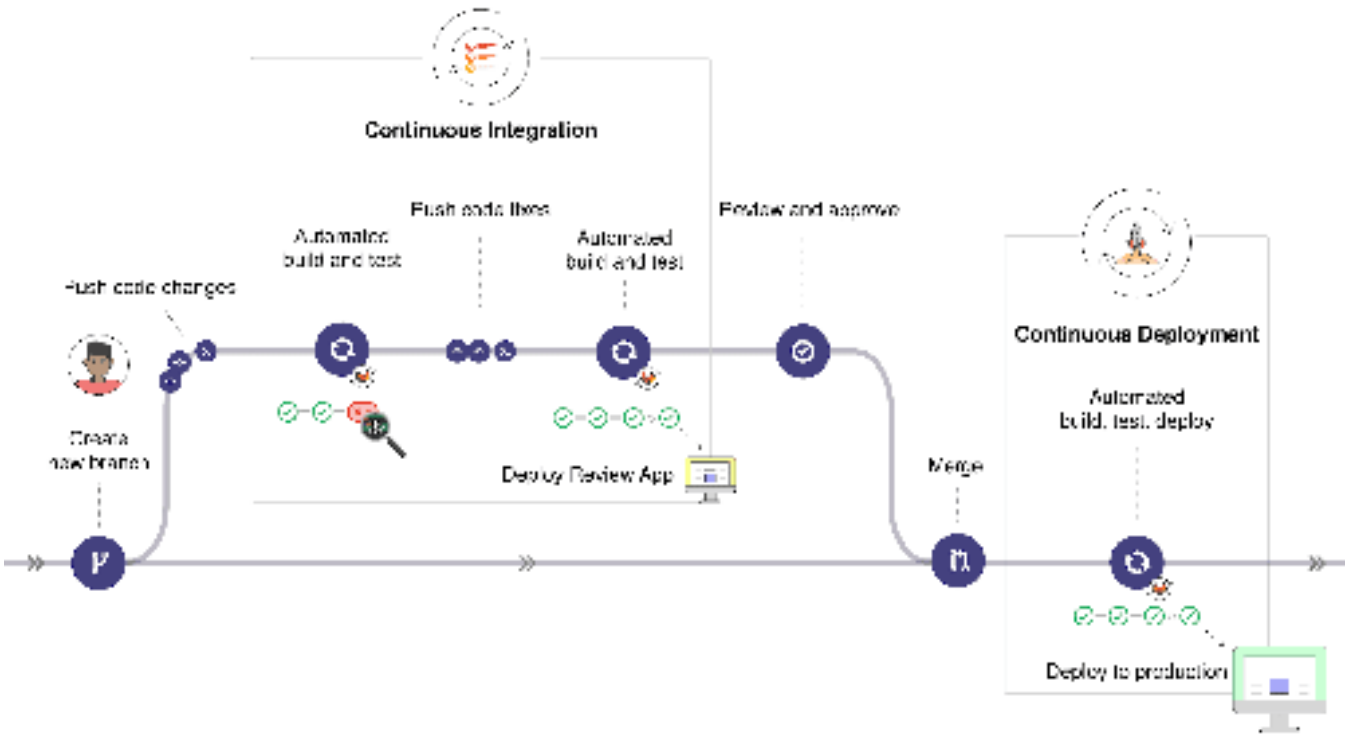


Figure 29: GitLab CI/CD workflow

The following figures highlight an initial implementation and test run of a configured *pipeline* on top of a hosted example project written in Python. The directed acyclic graph (DAG) of the *pipeline* is presented in Figure 30, whereas in Figure 31 the jobs view with more details is presented.

The *pipelines* and jobs are executed by a native GitLab runner, installed alongside GitLab, in a dedicated project server. The overall NESTLER integration deployment is following the cloud-native paradigm, hence, NESTLER components get delivered (deployed) using relevant standard procedures. These procedures include building docker images of the software and uploading them to a private container registry, testing the uploaded image and, when it comes to tags, deploying them.

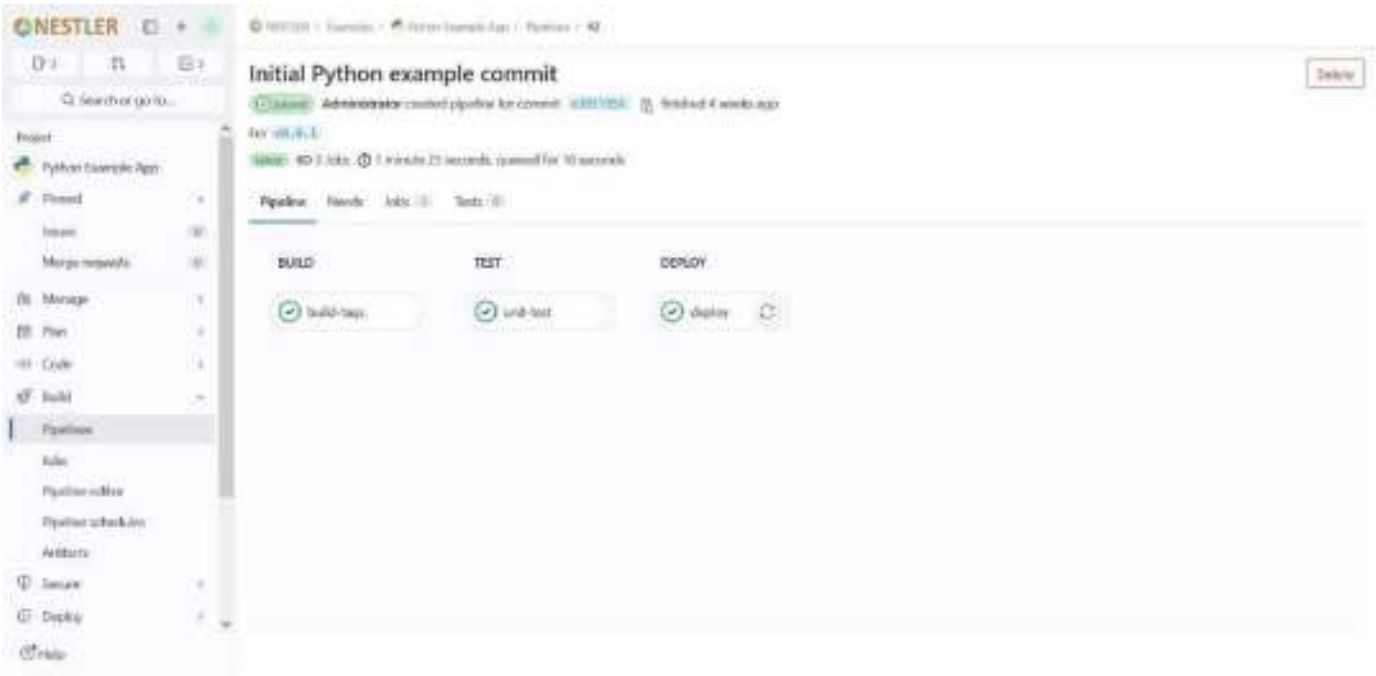


Figure 30: CI / CD Pipeline view of a NESTLER's project

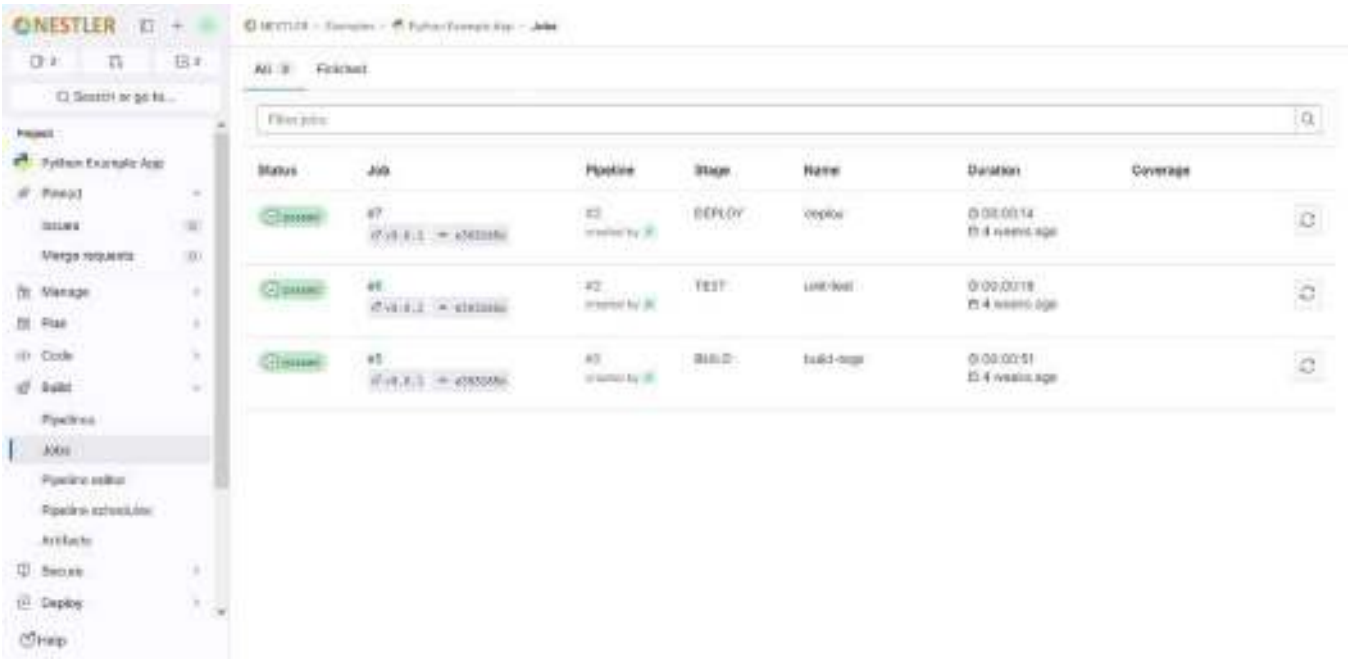


Figure 31: CI / CD Job view of a NESTLER's project

In the following listing, the contents of an indicative ".gitlab-ci.yml" file are presented. The stages of this particular pipeline are BUILD, TEST, SAST and DEPLOY, and each of them is composed of one or more *jobs*. For instance, the BUILD *stage* is composed of two build *jobs*, i.e. *build-main*, which is executed when a branch is merged or a commit is pushed directly to the main branch, and *build-tags*, which is executed only when release tags are created. The other *stages* of this pipeline are composed

by a single job each, and refer to unit-testing (TEST), static application security testing (SAST) and actual deployment to the deployment infrastructure (DEPLOY).

As it may be observed in the listing, deployment is taking place in Kubernetes. For this purpose, a GitLab agent has been deployed and configured alongside GitLab, to handle the Kubernetes cluster integration, connectivity, and management. Some additional details about the deployment platform can be found in the next paragraph.

```
image: docker:24.0.5

include:
  - template: Security/SAST.gitlab-ci.yml

services:
  - docker:24.0.5-dind

stages:
- BUILD
- TEST
- SAST
- DEPLOY

build-main:
  stage: BUILD
  before_script:
    - echo $CI_REGISTRY_PASSWORD | docker login -u $CI_REGISTRY_USER --password-stdin $CI_REGISTRY
  script:
    - docker build -t $CI_REGISTRY_IMAGE:latest .
    - docker image push $CI_REGISTRY_IMAGE:latest
  only:
    - main

build-tags:
  stage: BUILD
  before_script:
    - echo $CI_REGISTRY_PASSWORD | docker login -u $CI_REGISTRY_USER --password-stdin  $CI_REGISTRY
  script:
    - docker build -t $CI_REGISTRY_IMAGE:$CI_COMMIT_TAG -t $CI_REGISTRY_IMAGE:latest .
    - docker image push $CI_REGISTRY_IMAGE:$CI_COMMIT_TAG
    - docker image push $CI_REGISTRY_IMAGE:latest
  only:
    - tags

unit-test:
  stage: TEST
  script:
    - echo "Running unit tests... This will take about 10 seconds."
    - docker run $CI_REGISTRY_IMAGE:$CI_COMMIT_SHORT_SHA /script/to/run/tests
    - sleep 10
    - echo "Tests passed succesfully!"
```

```
sast:
  stage: SAST

deploy:
  stage: DEPLOY
  image:
    name: bitnami/kubectl:1.26.10
    entrypoint: ['']
  before_script:
    - sed -i "s+REGISTRY_IMAGE:REGISTRY_IMAGE_TAG+$CI_REGISTRY_IMAGE:$CI_COMMIT_TAG+"
deploy/kubernetes/deployment.yaml
  script:
    - kubectl config get-contexts
    - kubectl config use-context nestler/gitlab-agent:nestler-agent
    - kubectl apply -f deploy/kubernetes/pvc.yaml
    - kubectl apply -f deploy/kubernetes/configmap.yaml
    - kubectl apply -f deploy/kubernetes/secret.yaml
    - kubectl apply -f deploy/kubernetes/deployment.yaml
    - kubectl apply -f deploy/kubernetes/service.yaml
  only:
    - tags
```

4.4 Issue Tracking System

The term Issue Tracking System describes a software package that supports the creation and management of issues, and thus, the communication among a collaborating team. An issue could refer to a software bug, a new required or desired feature, functionalities that do not work as expected, etc. Besides the overall issue management and tracking, an Issue Tracking System offers a variety of functionalities that facilitate project planning, such as resource assignment, prioritization of tasks, time constraints, communication among collaborating people and / or teams, notifications, etc.

Some of the popular and widely used Issue Tracking Systems are Jira [55], Redmine [56], Bugzilla [57], etc. However, it is quite common for issue tracking to be handled by functionality inherent in the selected DevOps platform, which in the case of the NESTLER project is GitLab. GitLab indeed offers an advanced and feature-full tool for tracking the evolution of a new idea or the process of problem solving, as part of the GitLab workflow. A snapshot of NESTLER GitLab issue tracker is shown in Figure 32. The figure depicts the issues of the NESTLER group; it is given that issues can be managed on a per project basis as well.

Figure 32: Snapshot of NESTLER's group issues in GitLab

4.5 Containerization tools

Containerization tools, the major companion and competitor of virtualization, is becoming an increasingly popular, rapidly maturing technology among software development teams, as a facilitator of operations within the DevOps lifecycle. Containerization offers a higher layer of abstraction, over the application layer which packages code and library dependencies. The innate portability of containers among different computing environments has been one of the key features pushing the rapid development and adoption of containerization for a series of highly diverse applications. As it isolates the software from its execution environment, it ensures that the application works uniformly for a variety of systems, both bare-metal and OS-based.

The most common and popular containerization technology is Docker. A Docker container image is a lightweight, standalone, executable package of software that encapsulates the entirety of an application's prerequisites, such as code, runtime, system tools and libraries, etc.

Latest GitLab versions readily support the creation of a container registry, where Docker, or other types of container images can be pushed, pulled, and stored. The built-in GitLab container registry feature allows for a complete, self-hosted solution, without the need for deploying an additional, external registry for container images' storage purposes. Indeed, GitLab container registry has been utilised for NESTLER project and is currently hosted at:

https://registry.git.nestler-project.eu

A snapshot of the container registry, as captured via NESTLER's GitLab instance is depicted in Figure 33.

With the built-in container registry, a GitLab user does not need an account for an external registry (e.g. Dockerhub). Instead of this, the user can build, push, and pull container images to GitLab's container registry, given that their role within a project gives them sufficient privileges for such an operation. Typically, every user with a *Developer* or higher role can push and pull images to and from the container registry. The images of every project lie under the respective group and / or subgroup of the project. For instance, the *go-example-app* lies under the *Examples* subgroup of the *NESTLER*

group. The image is then accessible via *registry.git.nestler-project.eu/nestler/examples/go-example-app:<tag>*.

Figure 33: Snapshot of NESTLER's container registry

4.6 Deployment Platform

NESTLER components and their subcomponents are going to be deployed in a Kubernetes cluster hosted by SYN. In this context, a dedicated NESTLER namespace has been created within this cluster. Figure 34 shows the current deployments in the NESTLER namespace. Most of NESTLER's component are going to be deployed with the specified setup unless other additional needs emerge during the project's duration (e.g. the need for an additional namespace or other type of deployment practice). Even if more than one namespace is needed, the intercommunication and interoperability of NESTLER's subcomponents will be available via the standard visual network abstraction layer of Kubernetes, in combination with appropriate Kubernetes services for each of the components.

For the sake of completeness, it should be mentioned that resources with fundamental Kubernetes guidelines have been created in a dedicated project lying under the *NESTLER* group of the project's GitLab instance. Moreover, examples with respect to the appropriate setup for triggering GitLab CI/CD *pipelines* are included as well, lying under the *NESTLER/Examples* subgroup.

```
) get pods -n nestler -o wide
NAME                                                READY   STATUS    RESTARTS        AGE    IP              NODE              NOMINATED NODE   READINESS GATES
keycloak-0                                          1/1     Running   1 (2d16h ago)   3d21h  10.244.5.116    compute-r540-0    <none>           <none>
keycloak-postgresql-0                               1/1     Running   0               3d21h  10.244.3.98     compute-r540-3    <none>           <none>
minio-868cdf7b9f-lvqtf                              1/1     Running   0               3d18h  10.244.3.49     compute-r540-3    <none>           <none>
nestler-agent-gitlab-agent-v1-654b95f746-kn6kt      1/1     Running   5 (3d16h ago)   13d    10.244.6.6      compute-r540-1    <none>           <none>
nestler-agent-gitlab-agent-v1-654b95f746-qglrm      1/1     Running   0               13d    10.244.4.38     compute-r540-0    <none>           <none>
nestler-geoserver-master-0                          1/1     Running   19 (16h ago)    2d18h  10.244.5.99     compute-r540-0    <none>           <none>
nestler-geoserver-replica-0                         1/1     Running   2 (19h ago)     2d18h  10.244.5.100    compute-r540-0    <none>           <none>
nestler-geoserver-replica-1                         1/1     Running   0               2d18h  10.244.3.103    compute-r540-3    <none>           <none>
nestler-geoserver-replica-2                         1/1     Running   0               2d18h  10.244.6.139    compute-r540-1    <none>           <none>
nestler-postgresql-0                                1/1     Running   0               2d18h  10.244.3.99     compute-r540-3    <none>           <none>
```

Figure 34: An instance of the NESTLER namespace

Considering that data is of vital significance in the context of NESTLER, and the project is following the cloud-native paradigm, the existence of cloud-native storage is mandatory. On this ground, storage has been configured utilizing the open-source rook framework [58], combined with Ceph [59] as a software storage cluster. With this combination, effective and configurable storage duplication

can be achieved transparently and effortlessly for the end-user application, without the need to adhere to specified component design practices to support it.

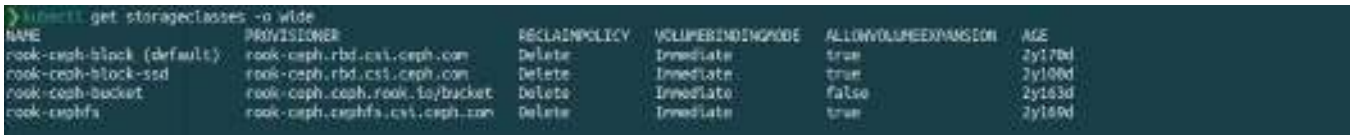


Figure 35: Storage classes of the Kubernetes cluster

As a final notice, since Kubernetes might not be available to all developers during the coding and debugging procedures, docker-compose versions of the deployments will be available to the component developers, for the local development to be transformed into an easier procedure, closer to the real environment, rendered possible by the utilization of containers for development as well as local testing.

5 NESTLER AI backend utilization

In the following sections, we provide some applications that in the forthcoming months will be tested and validated using the NESTLER AI Framework.

5.1 Weather impact assessment AI Algorithms

The NESTLER consortium members have identified drought and flood management as being a pressing challenge that farmers face in Africa.

5.1.1 Drought

Drought defines the state of an environment faced with a very long and significant lack of water so that it has impacts on flora, fauna and societies. Whenever it unexpectedly occurs, it causes declines in production that compromise food security, while the economic losses of the impacted countries are generally estimated in thousands of millions of US dollars.

The table below depicts some recent drought occurrences and the negative social and economic effects that it has on the corresponding countries.

Table 10: Some recent cases of drought and their social and economic effects

Country	Region	Negative effects	Year
Cameroon	Far North, North and Adamawa regions	Famine, disease, declines in production, disappearance of species, appearance of invasive species such as insect pests. Degradation of natural resources, displacement of populations, disruption of economic activities, especially agricultural and increasingly heavy economic and social costs, lack of drinking water, Losses are estimated at around $1.5 million US	2015
Ethiopia	Southern and eastern lowlands	More than 20 million people were affected, about 8 million faced food insecurity and acute malnutrition, including 2.9 million children and pregnant/ lactating women, water shortage (13 million in need of water), Psychological distress, migration.	2021
Rwanda	Whole country, specifically eastern and southern parts	Disruption of ecosystem equilibrium, loss of crop species, outbreak of new diseases and pests. Food insecurity, depletion of households' incomes, increase in malnutrition and health problems. Low crop productivity, decrease in produce quality, decreased incomes, depletion of households' incomes and national revenues.	2016

5.1.2 Flood

A flood is an overflow of water that submerges land that is usually dry and causes a lot of damage. Whenever it occurs unexpectedly, it can destroy thousands of hectares of cultivated lands and thousands of farm animals, thus compromising food security.

The table below depicts some recent flood occurrences and the negative social and economic effects that it has on the corresponding countries.

Table 11: Some recent cases of flood and their social and economic effects

Country	Region	Negative effects	Year
Cameroon	Far North,	More than 3 thousand destroyed infrastructures (houses, schools, hospitals, roads, and bridges), lower water quality, cause sedimentation and erosion. More than 3 million people were affected (injured, dead, homeless), loss of livelihoods. Fatality and damage; nearly 58,824 hectares of crops destroyed, including corn, several hundred head of cattle were killed and several fish ponds damaged.	2010 and 2020
Ethiopia	Afar, Gambella, SNNPR, Amhara, Somalia regions and Dire Dawa town	Between 2020 and 2022, 500,000 people have been affected and about 300,000 people were displaced due to flood in Ethiopia. Farmland and grazing land devastation through erosion and silt accumulation.	2012 and 2020
Nigeria	Low-lying areas and areas close to rivers or other bodies of water.	Floods can have negative impacts on the ecosystem, as they can disrupt the food web and affect other organisms that depend on the affected areas. Floods can have significant impacts on human populations, as they can lead to loss of life, displacement of people, and damage to homes and infrastructure. Floods can also lead to food shortages and economic losses. Can lead to damage to crops and infrastructure, as well as loss of income and economic activity.	2012 and 2020

5.1.3 Proposed solution

To manage drought and flood risks in region of medium and high risk, we have developed an AI based web app that can predict:

- whether the soil of a region will be in a state of drought or not.
- whether there will be a flood in a region or not.

To mitigate the problems caused by drought the web app provides predictions of whether a region will face a state of soil drought 30 to 90 days in advance.

To mitigate the problems caused by flood the web app provides predictions of whether a region will face flood during the rainy season due to excess precipitation. Thanks to this app, farmers can anticipate drought and flood and take necessary measures to control their impacts.

5.1.4 Scope

Within NESTLER, we aim to support drought prediction for the following regions of Cameroon:

- Balaji;
- Banyo;
- Batao;
- Belel;
- Guider;
- Kaele;
- Meiganga
- Lam;
- Tibati
- Toubouro.

Moreover, we aim to support flood prediction only for the five following regions of Cameroon:

- Guider;
- Kaele;
- Tibati
- Toubouro;
- Yagoua.

After successful testing for Cameroon, the goal will be to add regions of Ethiopia, Nigeria and Rwanda where there are high or medium risks of drought or flood.

5.1.5 Implementation

The following subsections give to the reader an overview of the implementation of the algorithms.

5.1.5.1 Datasets

We extracted our dataset from Power Data Access Viewer which is a global historical weather data viewer developed by NASA.

To develop our Drought AI/ML models, we used **daily precipitation, temperature and humidity** per region data from January 1983 to December 2022. We labelled the dataset based on the information provided by the Global SPEI Database website.

To develop our Flood AI/ML models, we used a dataset of **daily precipitation** per region data from January 1981 to December 2022. We leveraged news reports on flood events in Cameroon which are available online to set the monthly precipitation thresholds that we can use to label the dataset.

5.1.5.2 AI/ML models

We trained a Randomforest classifier with weather data from NASA website to obtain our prediction models for drought. For our prediction models for flood, depending on the region we used different classifiers (Randomforest, Logistics regression, Decision tree, Extra decision tree and Support vector machine) based on the performance on the dataset of the respective region. More information and specific UI will be provided in the next iteration of the deliverable.

Figure 36 depicts the use case diagram of the web app.

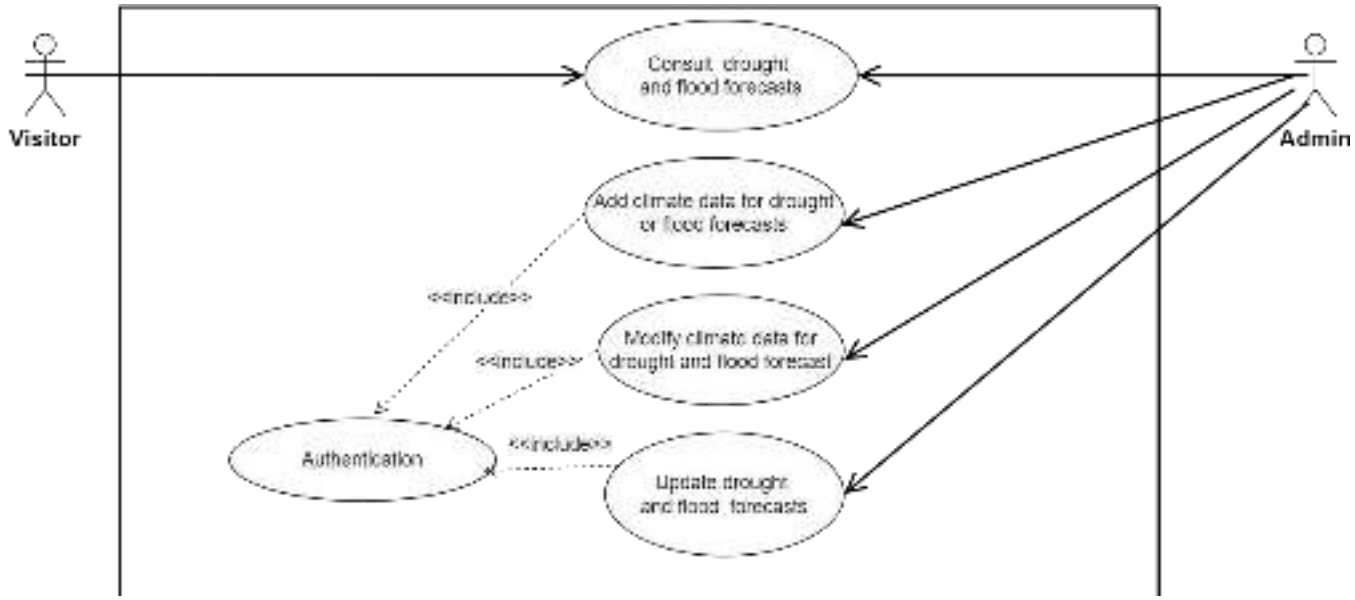


Figure 36: Weather prediction Use case diagrams

Figure 36 depicts the sequence diagram of the web app.

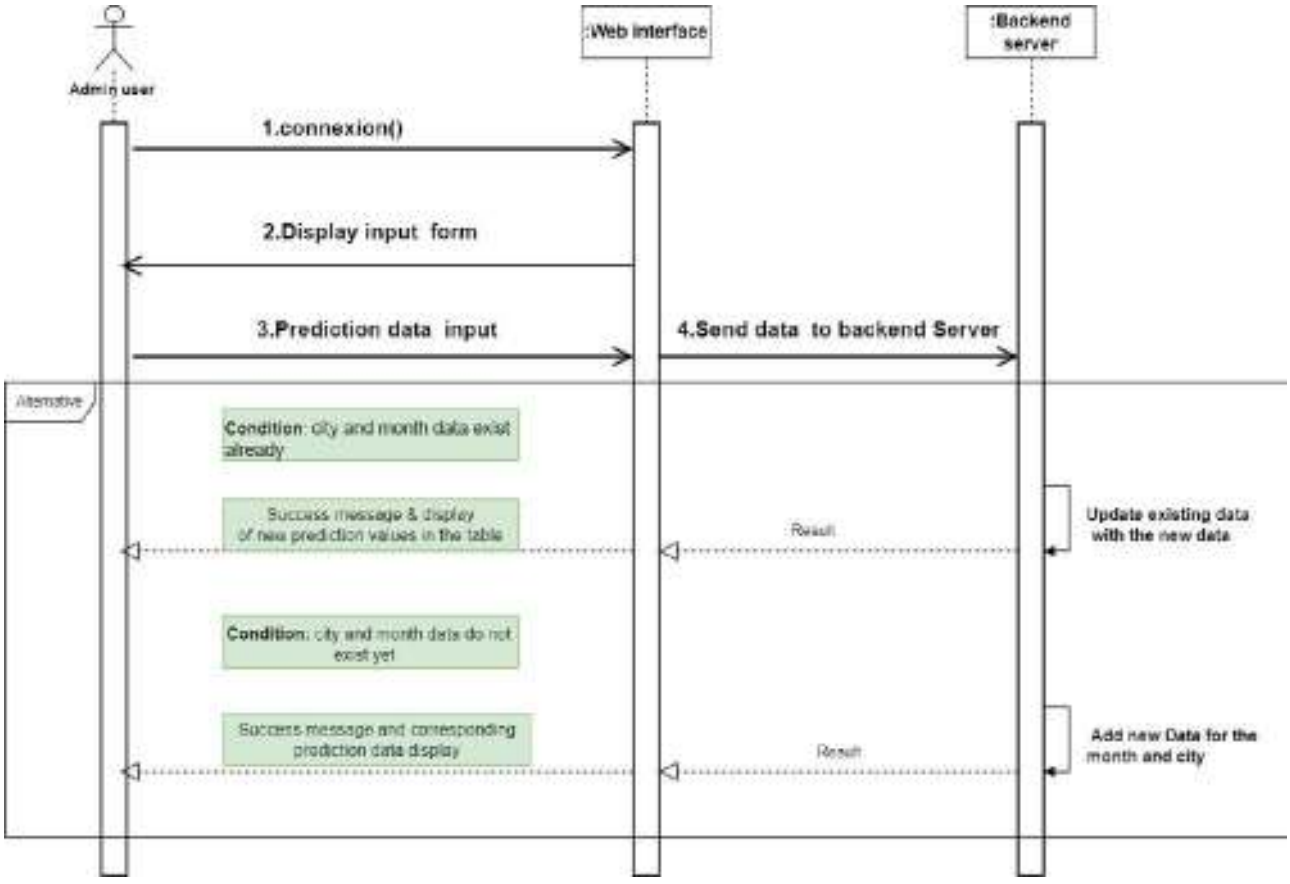


Figure 37: Sequence diagrams

5.2 Smart irrigation and pest repelling solutions

A European Parliament report accounts that farming activities consume almost 70% of global water withdrawals, reaching as much as 95 % in some developing countries, while about 13,000 to 16,000

litres of water are needed to produce 1 kg of beef, depending on the production system and the feed. Spain, Italy, Greece and France together account for 88% of the total EU irrigation water[1]

Figure 38: Volume of water for irrigation in EU

In Africa, of the total amount of water withdrawn, 85% is used in agriculture, 9% for community use and 6% for industry[2]. As it is shown in Figure 39, the problem is more important in the North Africa region, however regions such as Cameroon are vulnerable.

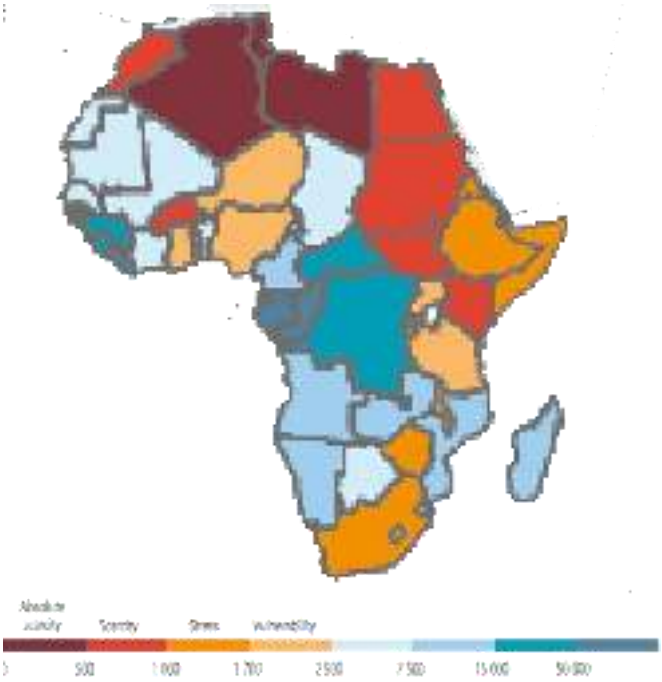


Figure 39: Total renewable water resources per capita

The NESTLER consortium have identified farm irrigation and pest control as being pressing challenges that farmers face in Africa.

5.2.1 Smart irrigation solution

To smartly control water usage, we have developed a smart irrigation solution. Environment and weather sensors continuously monitor and return environment and weather conditions to the NESTLER platform through the local node installed in the farm. The NESTLER platform then processes the received data and determines if the irrigation system has to be started or stopped by opening or

[1] https://www.europarl.europa.eu/RegData/etudes/BRIE/2019/644216/EPRS_BRI(2019)644216_EN.pdf

[2] United Nations, Sustainable Land and Water Management (SLWM) including Integrated Watershed Management Stratégies to ensure Food Security in Africa

closing the water source supplying the system. The opening and closing of the water source is controlled by a solenoid valve which receives its orders from the NESTLER platform via the local node. The NESTLER smart irrigation system is composed of:

- SynField X3 and peripheral nodes to control the sensors and actuators in the farm;
- Weather ISS including a rainfall sensor, a temperature and humidity sensor, a wind sensor and a pyranometer;
- a soil moisture sensor 10HS
- Multiple pulse-driven solenoid valves.

The solution is currently under laboratory test to ensure optimal irrigation procedures and will be installed and validated in Camerron in Q4 2024.

5.2.2 Smart pest detection and repelling solution

To smartly control pest repellents usage, we have developed an AI based mobile app which is a plant disease and pests diagnosis tool. It allows farmers to identify plant diseases or pests based on an image of a plant taken with the camera of his smartphone. It then provides treatment recommendations: type and quantity of pest repellents to use on the attacked farm.

This mobile app can run offline allowing farmers in areas without network connection to use it. It is a light app that can run on basic android smartphones such as Android 8 and more. The AI models that have been trained to identify pest infestations are integrated into it. This integration allows users to harness the power of advanced machine learning techniques for on-the-spot pest detection and management in agricultural settings.

Figure 40: NESTLER mobile application to identify disease on a tomato plant

5.3 Automated monitoring for crop yield quality

This task aims to develop algorithms that can process high-speed, high-volume signals captured by NESTLER sensors, to develop temporal association models that enable cross-correlation among distributed sensor networks, which include the data collected from earth observation repositories, environmental sensors, and other forms of devices (IoT, cameras, etc.), and to develop algorithms and methodologies that effectively combine disparate knowledge resources into a single framework leveraging upon the information fusion algorithms reported in the literature.

Currently, an automated algorithm for crop yield quality estimation and monitoring has been developed and tested in a region in Uganda, using as input satellite imagery and actual crop yield data for sorghum - the latter for correlation and evaluation purposes.

5.3.1 Algorithm for crop yield quality

Automated monitoring can provide real-time data on plant health and environmental factors. This information can be used to optimize irrigation and fertilization, reduce risks, and improve crop yields. The implementation of automated crop yield monitoring services in Sub-Saharan Africa addresses several critical issues faced by the region.

- **Food Security**: Reliable yield predictions assist in managing food supplies, helping to mitigate the risks of famine and food shortages.
- **Agricultural Productivity**: Such services enable farmers and agricultural agencies to monitor crop health in real-time, allowing for timely interventions to improve crop management and increase productivity.
- **Resource Optimization**: Accurate yield forecasts help in the efficient allocation of resources like water, fertilizers, and seeds, promoting sustainable agricultural practices.
- **Economic Stability**: By providing early warnings of potential yield changes, these services support economic planning and stability in a region where agriculture is a key economic sector.
- **Adaptation to Climate Variability**: With the increasing impact of climate change, real-time monitoring aids in adapting farming practices to varying climatic conditions, thereby reducing farmers' vulnerability to weather extremes.

5.3.2 Proposed solution

Within NESTLER, we propose to use an innovative solution for real-time crop yield monitoring in sub-Saharan Africa developed by Lillian Kay Petersen: "Real-Time Prediction of Crop Yields From MODIS Relative Vegetation Health" [60]. The approach utilizes MODIS satellite data to analyze vegetation health through the interpretation of NDVI, EVI, and NDWI indices anomalies on a monthly basis. The strength of this method lies in its broad applicability to different crops, climates, and locations, making it especially relevant for the diverse agricultural landscapes of Africa.

The proposed solution has demonstrated promising results in initial tests conducted in Illinois, USA, showing high accuracy in predicting yields of corn, soybeans, and sorghum. Its subsequent application across Africa yielded impressive results, with a substantial proportion of predictions achieving remarkably low error rates. Notably, this method does not require specialized crop masks, extensive tuning, or detailed subnational ground truth data, enabling its deployment in areas with minimal agricultural data infrastructure.

5.3.3 Algorithm workflow

The proposed crop yield quality algorithm uses MODIS satellite data, which is processed and analyzed through a combination of database management, cloud filtering, vegetation index calculation, and statistical analysis. MODIS satellite data are retrieved via an API from a selected platform, possibly including resources like NASA's MODIS Web Service API. After, data are processed (Cloud masking, Indices, Correlations) using the open-source spatial database, PostGIS. Monthly, the Vegetation Index

(VI) is generated. The final step involves transferring this data to a WebGIS platform for visualization. This will allow users to access updated and predicted crop yields for different regions, providing valuable insights.

The workflow is as follows:

1. **Data Acquisition and Processing**:
    - **Retrieval of MODIS Data**: MODIS satellite data, which is available twice daily, are obtained through an open-source platform.
    - **Database Management**: The retrieved data are systematically stored and processed in a database, using PostgreSQL DataBase Management System (DBMS).
2. **Cloud Masking and Data Cleansing**:
    - **Application of Standard MODIS Cloud Mask**: To ensure data quality, the Standard MODIS Cloud Mask identifies and excludes pixels affected by clouds or snow.
    - **Exclusion of Heavily Clouded Images**: Images with excessive cloud cover are excluded from the analysis to maintain data integrity.
3. **Calculation of Vegetation Indices (VIs)**:
    - **Monthly Average Computation**: Vegetation indices, including Normalized Difference Vegetation Index (NDVI), Enhanced Vegetation Index (EVI), and Normalized Difference Water Index (NDWI), are calculated based on monthly averages. This calculation excludes pixels identified as clouded or snow-covered.
    - **Creation of VI Climatology per Pixel**: A long-term average (climatology) of each VI for each pixel is computed using historical MODIS data. This climatology serves as a baseline to compare against the current or specific year's vegetation health.
4. **Correlation and Regression Analysis**:
    - **Correlation with Key Crops**: The system analyses the correlation between the VIs (both averages and anomalies) and the yields of the most important crops in each region.
    - **Linear Regression Analysis**: A linear regression is performed between the month with the highest NDVI value and crop production to predict yields.
5. **Geospatial Visualization in WebGIS**:
    - **Conversion to Vector Format**: VI anomalies and crop yield predictions are converted into a vector format for ease of visualization.
    - **WebGIS Integration**: The processed data will be displayed in a WebGIS platform, allowing users to visualize crop yield estimations and predictions during the year of harvest.

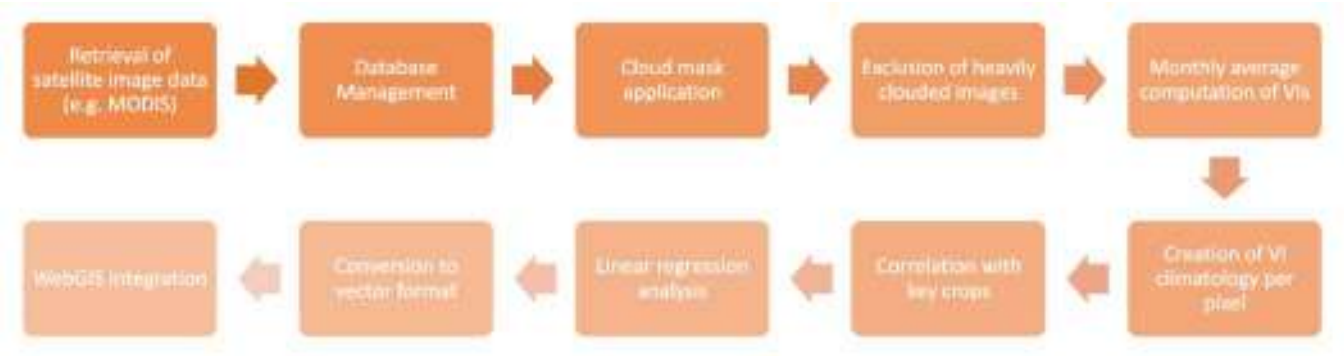


Figure 41: The workflow for processing satellite image data and integrating it into a WebGIS system for the crop yield quality algorithm.

5.3.4 Case study

A use case was undertaken for a region in Uganda to analyze and enhance agricultural decision-making through MODIS data analysis. MODIS satellite data were retrieved and analyzed for the years 2012 up to 2019, in order to calculate the averaged Vegetation Index (VI) and establish correlations with actual sorghum crop yields during the specified years.

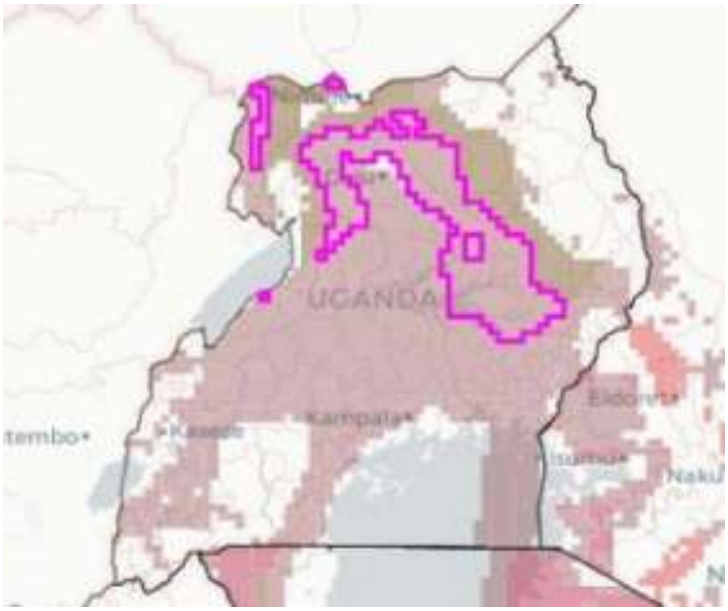


Figure 42: Map of Uganda and highlighted area for the case study

The data cover the years between 2012 and 2019, providing in this way a 7-year span with insights into vegetation patterns and changes over time. Vegetation indices (VI) were calculated, according to the above-mentioned methodology. Averaging the VI values across the chosen years allowed the identification of trends and anomalies in vegetation growth and thus crop yield estimation.

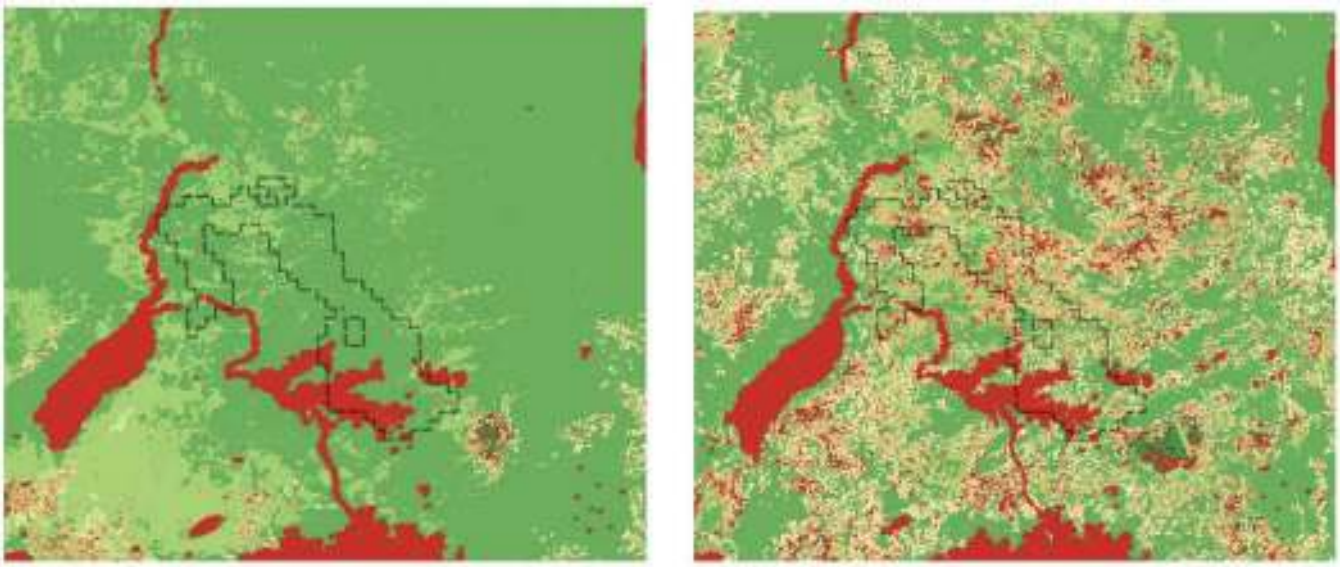


Figure 43: Visual example of Vegetation Index for January 2013 (left) and April 2013 (right).

To reveal meaningful insights into the relationship between vegetation health and crop productivity, a correlation between the averaged VI and actual sorghum crop yields in Uganda during the study period was calculated. For the specific time range, actual crop yield values of sorghum were used. Specifically for the study area, in 2013, crop yield was equal to 0.8 tonnes/hectare/harvested area, in 2015 was equal to 1.1 tonnes/hectare/harvested area, and in the year 2019, crop yield was equal to 0.9 tonnes/hectare/harvested area. A correlation value of -0.69 was calculated, indicating a noteworthy relationship between the extracted VIs and the corresponding crop yield.

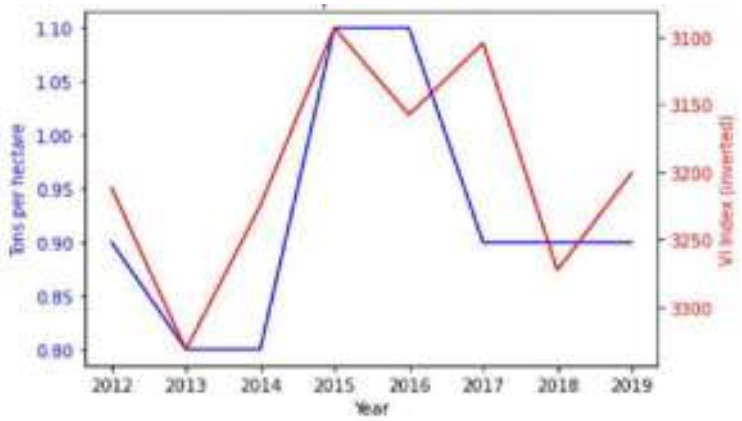


Figure 44: Correlation between the actual yield and the VI Index (inverted) over the years.

The next steps of this task involve the following activities:

- For the crop yield quality algorithm development:
  - to validate the model with more data
  - to create models for other crops and compare the results
  - to perform analysis in other countries with good crop yield data
- Algorithm integration regarding Livestock wellbeing and Insect population

5.4 Economic risk assessment models for predicting the yield quality

The integration of advanced technologies and data-driven methodologies in the field of agriculture can contribute in addressing complex challenges that can occur due to climate change and economic factors.

Task 4.3 aims to develop an economic risk assessment model based on already published models and taking into account factors such as the expansion of the agricultural land and the intensification of crop yields. Given the rise in global trade volume for agricultural products, there is a significant nutritional interdependency by society on food products from various regions. However, climate change is expected to impact the distribution of agricultural production, subsequently influencing food supply and global markets. In order to address this complex dynamic, the task includes training algorithms utilizing deep-learning techniques to predict the risk index, employing methods such as regression.

5.4.1 Economic Risk related definitions

In order to define the final outcome, some basic definitions are provided:

- ***Risk*** is the concept of future uncertainty associated with deviations from expected earnings or results. it represents the degree of uncertainty an investor is willing to accept to make a profit from their investment. In terms of economics, risk is the possibility that unexpected events may impact the intended results of economic actions, resulting in financial losses or missed opportunities. [61].
- ***Economic risk*** is the probability that changes in macroeconomic conditions will have a detrimental effect on a firm or investment. For example, changes in exchange rates or political unrest may affect profits or losses. In reality, the term "economic risk" is most frequently used to refer to the possible dangers of investments in foreign markets, particularly ones with political instability or poor government performance [62].
- ***Risk quantification*** involves the systematic analysis of identified threats and the generation of the necessary information to decide what actions should be taken [63]. There are two major methodologies for risk quantification: qualitative and quantitative approaches: a) ***Qualitative approaches***, such as assessing a company's overall strategy, are subjective and managed as part of the business decision-making process and b) ***Quantitative risk assessment*** involves defining the probability of occurrence and potential impact of each type of risk important to the organization's business. Quantitative assessment often utilizes statistical models, Monte Carlo simulations, time series models, optimization, machine learning, and other computational techniques (*Evaluating Risks Using Quantitative Risk Analysis*, n.d.).
- ***Risk prediction models*** are statistical models or tools that aim to forecast the likelihood or probability of a future event. These models combine information from various attributes to help identify individuals at a higher risk of encountering specific negative outcomes, such as developing a particular disease or experiencing adverse events [64]. The two primary categories of such models are statistical models, which include methods like regression analysis to estimate event likelihood, and machine learning (ML) models that learn from data to generate predictions [65].

5.4.2 Risk assessment examples

As an example of developing a risk assessment model we may consider the risk prediction models which estimate the likelihood of future outcomes based on individual characteristics [66]. The focus is on how researchers develop and validate these models using individual participant data (IPD) meta-analysis. The methods involve a qualitative review of 15 articles, examining the aims, methodology,

and reporting of risk prediction models developed from IPD across multiple studies. The results highlight the opportunities and challenges of the IPD approach, including issues with data availability, missing information, and heterogeneity in methods among studies. Most articles perform internal validation, limiting generalizability, while only a few use external validation. The study emphasizes the unique opportunities of IPD meta-analysis for risk prediction research, suggesting improvements in model generalizability and validation methods, along with advocating for planned collaborations to address methodological challenges.

According to another study of developing an economic risk assessment model [67], enhancements in forecasting datasets reveal limitations in traditional models, prompting an evaluation of machine learning methods for predicting monetary policy actions and associated macroeconomic risks. By expanding the information set on Chinese systemic risk, the results confirm its value for macroeconomic forecasting. Notably, machine learning processes, particularly quantile regression forest, demonstrate superior out-of-sample prediction accuracy compared to conventional methods. These findings hold significance for policymakers and investors alike.

5.4.3 Data Acquisition

The development of the Economic Risk Assessment Model relies on the accurate selection and acquisition of datasets. The primary dataset was obtained from the Food and Agriculture Organization (FAO) of the United Nations[3] and it was chosen for its comprehensive agricultural data. This dataset includes critical attributes, focusing on specific African countries, distinct agricultural products for a period of 30 years from 1990 to 2020. In particular, the dataset includes information about Uganda, Kenya, Cameroon, Ethiopia, Rwanda and Nigeria and the 7 crops of interest that are cassava, coffee, banana, cowpea, soybean, yam and maize.

The selected attributes were carefully chosen to address key factors influencing the agricultural landscape, including the expansion of agricultural land area, intensification of crop yields, global trade volume, regional variations in food production, and the impact of climate change on agricultural distribution and global markets. In order to enrich the dataset's depth, multiple datasets were combined aiming to capture the complexity and interconnected nature of factors influencing economic risks in agriculture.

5.4.4 Filtering/Data Cleaning

To enhance the dataset's quality, columns or rows with a high rate of missing values (>55%) were removed from the dataset. The percentage was defined based on experimentations. Also, the datasets were organized based on specific agricultural products to find correlation between the attributes.

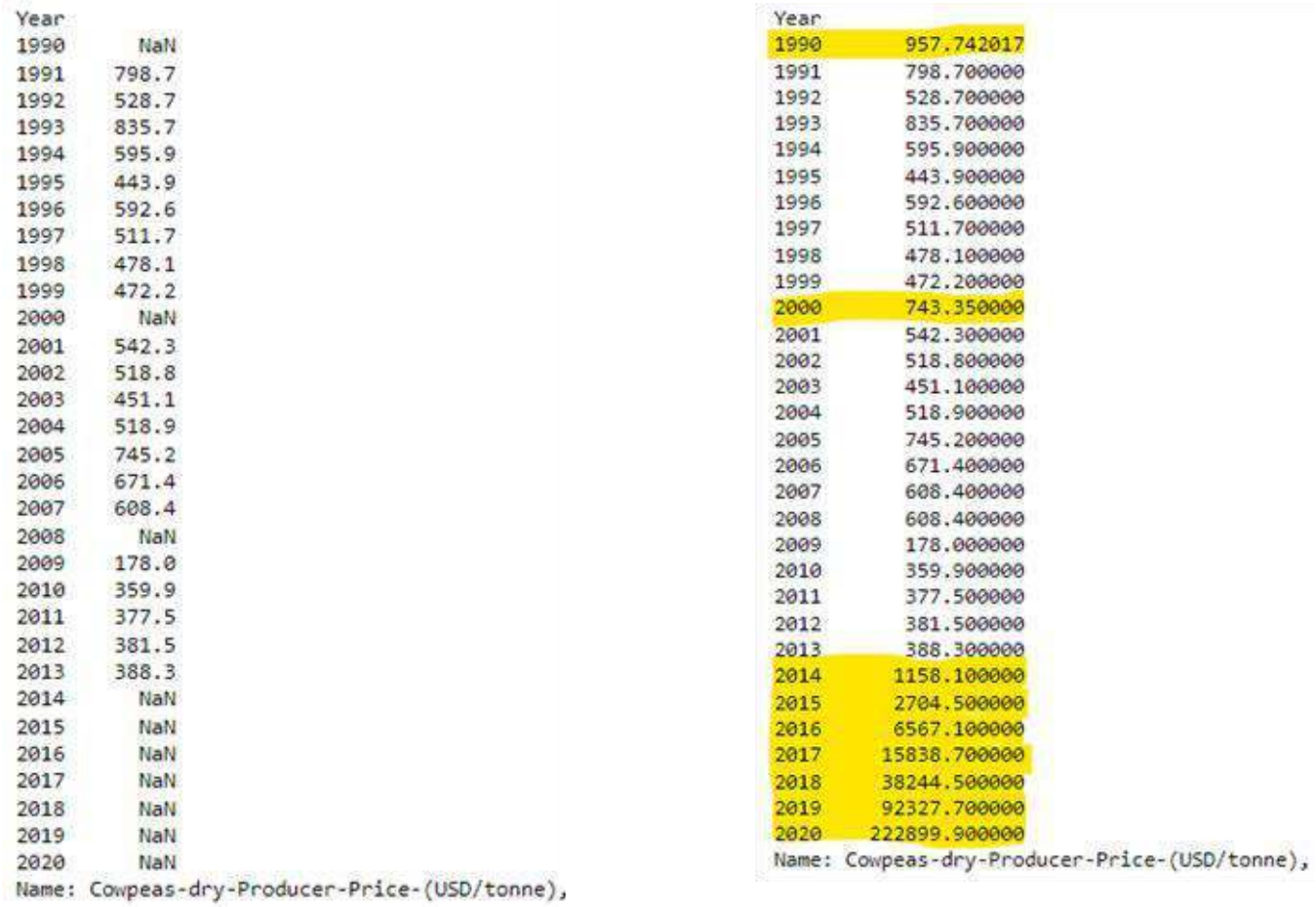
5.4.5 Fill in Missing Values

Addressing missing values is crucial for the preparation of the dataset, therefore, two approaches were considered: imputation and interpolation. Imputation involves replacing missing data with the column's mean, and interpolation is an estimation technique generating data points within the range of existing data points.

[3] https://www.fao.org/faostat/en/#data/QCL

5.4.6 Method 1: Least Square Approximation (LSA)

The usual approach for LSA is to minimize the sum of the squares of the errors for a polynomial of given degree, thus the “least-squares” principle [68]. Initially, it was attempted to interpolate all the missing values which resulted in very high numbers since the polynomial was a degree of more than 2 (*Figure 45*) The polynomial then gave non-logical values, thus any degree higher than three was rejected. As a result, this method is rejected for degree higher or equal than 2.



Year	
1990	NaN
1991	798.7
1992	528.7
1993	835.7
1994	595.9
1995	443.9
1996	592.6
1997	511.7
1998	478.1
1999	472.2
2000	NaN
2001	542.3
2002	518.8
2003	451.1
2004	518.9
2005	745.2
2006	671.4
2007	608.4
2008	NaN
2009	178.0
2010	359.9
2011	377.5
2012	381.5
2013	388.3
2014	NaN
2015	NaN
2016	NaN
2017	NaN
2018	NaN
2019	NaN
2020	NaN

Name: Cowpeas-dry-Producer-Price-(USD/tonne),

Year	
1990	957.742017
1991	798.700000
1992	528.700000
1993	835.700000
1994	595.900000
1995	443.900000
1996	592.600000
1997	511.700000
1998	478.100000
1999	472.200000
2000	743.350000
2001	542.300000
2002	518.800000
2003	451.100000
2004	518.900000
2005	745.200000
2006	671.400000
2007	608.400000
2008	608.400000
2009	178.000000
2010	359.900000
2011	377.500000
2012	381.500000
2013	388.300000
2014	1158.100000
2015	2704.500000
2016	6567.100000
2017	15838.700000
2018	38244.500000
2019	92327.700000
2020	222899.900000

Name: Cowpeas-dry-Producer-Price-(USD/tonne),

Figure 45: LSA applied on Cowpeas producer price dataset

5.4.7 Method 2: Cubic Spline Interpolation (CSI)

A common numerical curve fitting strategy is a cubic spline where it fits a "smooth curve" to the known data, using cross-validation between each pair of adjacent points in order to set the degree of smoothing and estimate the missing observation by the value of the spline. Splines can be of any degree, however, cubic splines are the most popular. CSI emerged as a successful method, utilizing a combination approach with a moving window and addressing challenges related to missing values close to each other.

5.4.8 Method 3: Moving Average (MA)

Moving average is another method for missing value imputation where missing values are replaced by weighted moving average. MA proved effective, utilizing a moving window that fills missing values with the mean. The approach considered the quantity of missing values in each half of the time series

[69]. The above methods collectively contributed to generating a complete dataset, ensuring its integrity and reliability for subsequent analyses in the NESTLER project.

5.4.9 Statistical Analysis

Statistical Analysis constitutes a key stage in developing the Economic Risk Assessment Model. Initially, a comprehensive exploration of correlations is implemented, assessing time series similarity, and crafting the target column's formula. Then, visualization, outlier detection, and rescaling from 0 to 1 contribute to enhancing data understanding and model robustness. Additionally, cross-correlation, dendrogram analysis, and stationarity tests further refine the analysis, ensuring a comprehensive and rigorous approach. Regression models are then trained, tested and evaluated, solidifying the model's reliability.

5.4.10 Next steps

The next steps in developing the economic risk assessment model include a series of essential stages. First, statistical analysis will continue with visualization, detection of outliers, rescaling from 0 to 1, cross-correlation, dendrogram analysis and stationary test.

Next, the correlation among features will be explored and the similarity of time series will be assessed. Correlation analysis will show relationships between different attributes thus, providing more information on how changes in one variable may affect others. Understanding the similarity of time series data will help identify patterns and trends that contribute to the overall economic risk assessment.

Following the correlation analysis, the target column will be developed based on the identified correlations and the information gained from the dataset. This is a critical aspect of developing the economic risk assessment model. The target column represents the variable of interest, i.e. the risk index associated with economic and agricultural factors.

The next step involves the development of regression models which aim to establish a relationship between the target variable and the independent variables, thus, allowing the prediction of economic risk based on various factors. At this step, multiple regression models will be explored, each capturing facets of the complex interplay within the dataset. Once regression models are defined these need to be trained. Training involves optimizing the model to accurately capture the patterns present in the data.

The final step is the evaluation of the developed regression models. For this, evaluation methods such as Mean Squared Error (MSE), Root Mean Square Error (RMSE) and R-squared will be used to assess the model’s performance. This step will ensure that the risk assessment model is robust and reliable.

6 Initial NESTLER platform architecture

This section presents the NESTLER reference architecture as it has been evolved from the beginning of the project. Its development has been guided by the system's intended functionality and the identified user requirements. Additionally, both Functional and Non-Functional requirements, as detailed in D1.1 “Review of Historical Case-Studies on Adverse Events, (Non-)Functional Requirements” [70] have been considered. The proposed architecture is designed to be flexible, allowing for adjustments and updates, and scalable, ensuring it can adapt to increasing demands and evolving technologies.

6.1 High-level System Architecture

The high-level system architecture of NESTLER platform is structured into different layers, each one of which is responsible for specific functionalities within the platform. Figure 46 depicts the high-level NESTLER architecture. At the base of the system, there are various data sources, including IoT sensors, drone, cameras, microphones data and satellite Earth observation. The system has the ability to integrate and analyze a wide range of data inputs from different technologies, providing multi-modality characteristics to the system.

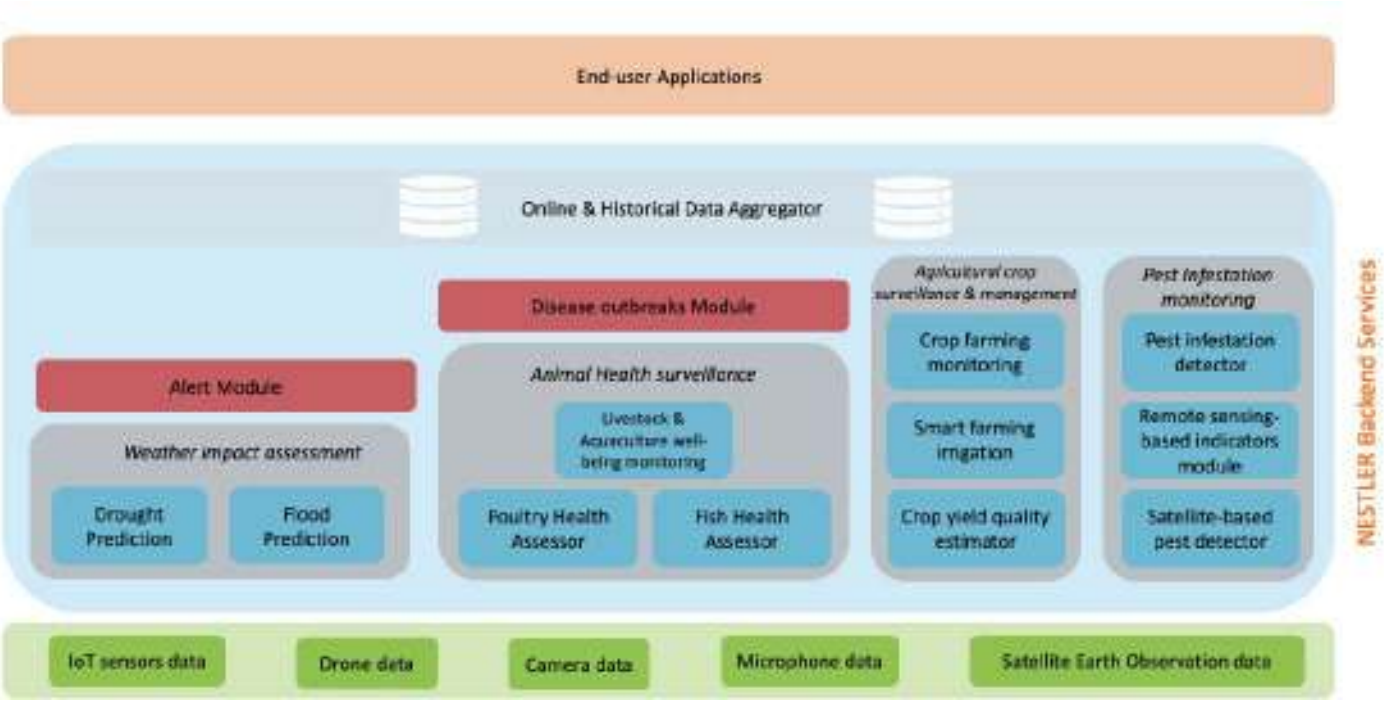


Figure 46: High-level reference architecture overview

6.2 Data sources

The description of the data, its acquisition methods through the NESTLER platform, and the platform's multimodal data aggregation characteristics are thoroughly detailed in D3.1 “Remote Sensing Technologies and Multimodal Data Aggregation Protocols” [71]. However, this section aims to offer a holistic overview of the high-level architecture of the NESTLER platform by outlining the various data sources that provide data to the system.

- *IoT sensors* are devices within the Internet of Things (IoT) ecosystem, designed to gather diverse types of data, which can be used for real-time monitoring and decision-making in services. In the NESTLER project, data is collected using the SynField platform, which supports connections with the following devices:

  - *Weather station* that offers remote monitoring of environmental weather factors.
  - *Soil sensors*, gathering data related to soil conditions.
  - *SynWater*, which provides measurements for water quality.
  - *SynAir*, monitoring air quality.
- *Quality Measurement Device* extracts measurements related to crop quality characteristics. In the NESTLER project, those devices are used to gather information about the quality of cassava crops.
- *Drone* is a dynamic data source withing the system, providing high-resolution aerial imagery and geographic information. In NESTLER project, drones will be utilized to collect data, providing insights about the biotic stress of crops.
- *Camera* captures visual information in the form of images and video. In the NESTLER project, camera is used to gather video data of livestock and aquaculture well-being as well as images of crops.
- *Microphone* captures surrounding information from the surrounding environment. In NESTLER, microphones are employed to monitor acoustic signals within livestock farming.

6.3 NESTLER Services

The multi-modal data is fed into the NESTLER backend, where it is stored in the NESTLER database through the *Historical Data Aggregator*. Additionally, this data is processed by various services within the platform. To provide a brief overview, the services can be grouped into 4 main categories:

- *Weather impact assessment*, which consists of the services of *Drought* and *Flood prediction*. The output of each service is an indicator that forecasts the likelihood of a specific undesired event (drought/flood) in the coming days. If the system predicts that the event will occur, the *Alert module* is triggered to inform the users about the upcoming event.
- *Animal health surveillance,* which is composed of three services. The first one is the *Livestock & Aquaculture well-being monitoring*, which outputs records of environmental conditions of the habitats. The other two are the *Poultry* and *Fish Health Assessors* services. The outputs of those two services are a single health index. Those values are processed by the *Disease Outbreak Module* to detect any disease outbreak.
- *Agricultural crop surveillance & management*, aiming to observe and manage crop cultivation activities and conditions to achieve optimal crop growth. This category contains the *Crop farming monitoring*, *Smart farming irrigation* and the *Crop yield quality estimator*.
- *Pest Infestation monitoring*, including the *Remote sensing-based indicators module, the Pest infestation detector,* and the *Satellite-based pest detector*. It should be mentioned that those services can operate independently. However, the *Remote sensing-based indicators module* and the *Pest infestation detector* have the capacity to be integrated into a single, unified service.

The outputs of the various NESTER services are stored in the NESTLER Backend. Moreover, the collected data as well as the services' results are made accessible to users through various End-user Applications. A detailed description of those applications will be described in D4.2 "NESTLER frontend implementation of GIS services" on M24. However, a summary of the applications is provided here.

- *Visualization tool* based on Geographic Information System (GIS), designed to provide users with geographically contextualized graphical representation of the collected data as well as

services' results, enabling them to gain insights and make decisions about crop conditions and livestock, aquaculture well-being.

- *Mobile Application*, which offers on-demand features for detecting pests on crops. This mobile app can be used by the users to perform real-time detection of pest infestation, visualizing the results provided by the *Pest infestation detector*.
- *External Services API* will be implemented to facilitate the connection between the NESTLER backend and external platform services.

7 Conclusion

This deliverable presents the NESTLER AI Framework (NAIF), that is designed to facilitate the deployment of various Federated Learning frameworks, ensuring a versatile and efficient environment for training models using Federated Learning. The Initial NESTLER backend implementation of AI algorithms developed in this deliverable serves as the foundation for the development of the NESTLER backend implementation of AI algorithms and agricultural services, which is a main component of the NESTLER platform.

Moreover, the D4.1 defines the NESTLER Integration framework and tools, which is designed to be a dynamic and scalable, incorporating a DevSecOps approach that includes Source Code Management, CI/CD practices, Issue Tracking, and Containerization. The framework is intended to evolve with the project, with ongoing updates and detailed documentation to support the integration of various NESTLER components. Finally, the flexible and scalable NESTLER reference architecture, shaped by user needs and both functional and non-functional requirements, is demonstrated, with particular focus on the integration of data sources and services.

This document will be updated as the project progresses and reaches its next milestones.

The final version of this document will be produced by the project team in month 30 (D4.3 NESTLER backend implementation of AI algorithms and agricultural services).

8 References

[1] Forrester, "Federated Learning: A Distributed Machine Learning Technique Without Data Aggregation," 2022. [Online]. Available: https://www.forrester.com/report/federated-learning-a-distributed-machine-learning-technique-without-data-aggregation/RES177214.

[2] H. McMahan, E. Moore and D. Ramage, "Communication-efficient learning of deep networks from decentralized data," in *20th International Conference on Artificial Intelligence and Statistics, AISTATS*, 2020.

[3] T. Li, A. K. Sahu and M. Zaheer, " Federated Optimization in Heterogeneous Networks," 2018. [Online]. Available: http://arxiv.org/abs/1812.06127.

[4] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie and R. Pedarsani, "FedPAQ: A Communication-Efficient Federated Learning Method with Periodic Averaging and Quantization," 2019. [Online]. Available: http://arxiv.org/abs/1909.13014.

[5] J. So, B. Guler and A. S. Avestimehr, "Turbo-Aggregate: Breaking the Quadratic Aggregation Barrier in Secure Federated Learning," *IEEE Journal on Selected Areas in Information Theory,* vol. 2, no. 1, p. 479–489, 2021.

[6] L. Liu, J. Zhang, S. H. Song and K. B. Letaief, "Client-Edge-Cloud Hierarchical Federated Learning," in *IEEE International Conference on Communications*, 2020.

[7] X. Yao, T. Huang and R.-X. Zhang, "Federated Learning with Unbiased Gradient Aggregation and Controllable Meta Updating," 2019. [Online]. Available: http://arxiv.org/abs/1910.08234.

[8] S. Ek, F. Portet, P. Lalanda and G. Vega, "A Federated Learning Aggregation Algorithm for Pervasive Computing: Evaluation and Comparison," in *IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2021.

[9] M. Mohri, G. Sivek and A. T. Suresh, "Agnostic federated learning," in *36th International Conference on Machine Learning, ICML 2019,*, 2019.

[10] "TensorFlow Federated: Machine Learning on Decentralized Data," [Online]. Available: https://www.tensorflow.org/federated.

[11] T. Ryffel et al., "A generic framework for privacy-preserving deep learning," [Online]. Available: http://arxiv.org/abs/1811.04017.

[12] "FATE: An Industrial Grade Federated Learning Framework," [Online]. Available: https://github.com/FederatedAI/FATE.

[13] D. J. Beutel et al., "Flower: A Friendly Federated Learning Research Framework," 2020. [Online]. Available: http://arxiv.org/abs/2007.14390.

[14] T. Ryffel et al., "Sherpa.ai: Federated Learning and Differential Privacy: Software tools analysis, the Sherpa.ai FL framework and methodological guidelines for preserving data privacy," [Online]. Available: https://arxiv.org/pdf/2007.00914.pdf.

[15] C. He et al., "FedML: A Research Library and Benchmark for Federated Machine Learning," [Online]. Available: http://arxiv.org/abs/2007.13518.

[16] "PaddleFL: Paddle Federated Learning," [Online]. Available: https://paddlefl.readthedocs.io/en/latest/.

[17] S. Caldas et al., "LEAF: A Benchmark for Federated Settings," 2018. [Online]. Available: http://arxiv.org/abs/1812.01097.

[18] G. A. Reina et al., "OpenFL: An open-source framework for Federated Learning," [Online]. Available: http://arxiv.org/abs/2105.06413.

[19] NVIDIA DEVELOPER, "NVIDIA FLARE," [Online]. Available: https://developer.nvidia.com/flare.

[20] "Pytorch," [Online]. Available: https://pytorch.org/.

[21] [Online]. Available: https://nvflare.readthedocs.io/en/main/user_guide/operation.html?highlight=prompt#admin-command-prompt.

[22] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Foundations and Trends in Theoretical Computer Science,* vol. 9, no. 3-4, p. 211–487, 2013.

[23] F. McSherry and K. Talwar, "Mechanism Design via Differential Privacy," pp. 94–103, 2008, doi: 10.1109/focs.2007.66.

[24] Y. Zhao and e. al., "Local Differential Privacy-Based Federated Learning for Internet of Things," *IEEE Internet of Things Journal,* vol. 8, no. 11, p. 8836–8853, 2021.

[25] J. Lee and C. Clifton, "How much is enough? Choosing ε for differential privacy," *Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics,* vol. 7001, p. 325–340, 2011.

[26] M. Abadi and et al., "Deep learning with differential privacy," in *ACM Conference on Computer and Communications Security*, 2016.

[27] P. Subramani, N. Vadivelu and G. Kamath, "Enabling Fast Differentially Private SGD via Just-in-Time Compilation and Vectorization," 2020. [Online]. Available: http://arxiv.org/abs/2010.09063.

[28] C. Zhao and e. al, "Secure Multi-Party Computation: Theory, practice and applications," *Information Sciences,* vol. 476, p. 357–372, 2019.

[29] O. Goldreich and Y. Oren, "Definitions and properties of zero-knowledge proof systems," *Journal of Cryptology,* vol. 7, no. 1, pp. 1-32, 1994.

[30] P. Mohassel and Y. Zhang, "SecureML: A System for Scalable Privacy-Preserving Machine Learning," in *IEEE Symposium on Security and Privacy*, 2019.

[31] N. Kilbertus, A. Gascón, M. Kusner, M. Veale, K. P. Gummadi and A. Weiler, "Blind justice: Fairness with encrypted sensitive attributes," in *35th International Conference on Machine Learning (ICML)*, 2018.

[32] Wikipedia, "Homomorphic encryption," [Online]. Available: https://en.wikipedia.org/wiki/Homomorphic_encryption. [Accessed 14 Dec 2023].

[33] N. Papernot, M. Abadi, Ú. Erlingsson, I. Goodfellow and K. Talwar, "Semi-supervised Knowledge Transfer for Deep Learning from Private Training Data," in *5th International Conference on Learning Representations, ICLR 2017*, 2017.

[34] L. Aslett, P. M. Esperanç and C. Holmes, "A review of homomorphic encryption and software tools for encrypted statistical machine learning," 2015. [Online]. Available: http://arxiv.org/abs/1508.06574. [Accessed 2023].

[35] N. Papernot, S. Song, I. Mironov, A. Raghunathan, K. Talwar and Ú. Erlingsson, "SCALABLE PRIVATE LEARNING WITH PATE," in *ICLR*, 2018.

[36] N. Papernot, S. Song, I. Mironov, A. Raghunathan, K. Talwar and Ú. Erlingsson, "Scalable private learning with pate," in *6th International Conference on Learning Representations, ICLR*, 2018.

[37] I. Mironov, "Rényi Differential Privacy," in *IEEE Computer Security Foundations Symposium,*, 2017.

[38] N. G. Papernot, "Privacy-and-Machine-Learning @ Www.Cleverhans," [Online]. Available: http://www.cleverhans.io/privacy/2018/04/29/privacy-and-machine-learning.html.

[39] NVIDIA FLARE, "provision.py," 2022. [Online]. Available: https://github.com/NVIDIA/NVFlare/blob/b799c15ed82da5df32c3bff00aa92c15f366fa83/nvflare/lighter/provision.py.

[40] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, J. Fernandez-Marques, Y. Gao, L. Sani, K. H. Li, T. Parcollet, P. Porto Buarque de Gusmao and N. D. Lane, "FLOWER: A friendly federated learning framework," 2022.

[41] Y. Pan, J. Ni and Z. Su, "FL-PATE: Differentially Private Federated Learning with Knowledge Transfer," in *IEEE*, 2021.

[42] H. B. McMahan, E. Moore, D. Ramage, S. Hampson and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," 2017.

[43] J. Kiefer and J. Wolfowitz, "Stochastic estimation of the maximum of a regression function," *The Annals of Mathematical Statistics,* pp. 462-466, 1952.

[44] H. Hu, Z. Salcic, L. Sun, G. Dobbie, & X. Zhang . Source inference attacks in federated learning. In 2021 IEEE International Conference on Data Mining (ICDM) (pp. 1102-1107). IEEE.

[45] Y. Wang, C. Si, and X. Wu, "Regression Model Fitting under Differential Privacy and Model Inversion Attack."

[46] G. Jocher, "YOLOv5," https://github.com/ultralytics/yolov5, 2020.

[47] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Doll{\'a}r and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*, Springer, 2014, pp. 740-755.

[48] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779-788.

[49] M. Everingham, L. Van Gool, C. K. Williams, J. Winn and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision,* vol. 88, no. 2, pp. 303-338, 2010.

[50] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," 2009.

[51] "NSL-KDD dataset," [Online]. Available: https://www.unb.ca/cic/datasets/nsl.html.

[52] IBM Cloud Education, "DevSecOps," IBM, 30 07 2020. [Online]. Available: https://www.ibm.com/cloud/learn/devsecops

[53] GitLab, „GitLab Continuous Integration & Delivery," [Online]. Available: https://about.gitlab.com/product/continuous-integration/.

[54] GitLab, „Introduction to CI/CD with GitLab," 2019. [Online]. Available: https://docs.gitlab.com/ee/ci/introduction/index.html.

[55] Atlassian, „Jira Software," 2023. [Online]. Available: https://www.atlassian.com/software/jira.

[56] Redmine, „Redmine," 2023. [Online]. Available: https://www.redmine.org.

[57] Bugzilla, „Bugzilla," 2023. [Online]. Available: https://www.bugzilla.org/.

[58] „Open-Source, Cloud-Native Storage for Kubernetes," [Online]. Available: https://rook.io.

[59] „Ceph - Home page," [Online]. Available: https://ceph.io.

[60] L. K. Petersen, "Real-Time Prediction of Crop Yields From MODIS Relative Vegetation Health: A Continent-Wide Analysis of Africa. Remote Sensing," vol. 10, no. 11, p. 1726, 2018.

[61] "What is Risk? Definition of Risk, Risk Meaning," The Economic Times, [Online]. Available: https://economictimes.indiatimes.com/definition/risk. [Accessed Nov. 2023].

[62] J. Chen, "Risk: What It Means in Investing, How to Measure and Manage It. Investopedia.," 2023. [Online]. Available: https://www.investopedia.com/terms/r/risk.asp. [Accessed Nov 2023].

[63] J. M. Quigley and S. Lauck, "Defining Risk Management - PM Tips. Project Management Tips," 2019. [Online]. Available: https://pmtips.net/article/defining-risk-management.

[64] "Assessing the Value of Risk Predictions Using Risk Stratification Tables.," 2023. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3091826/.

[65] P. Chitanand, " Risk Prediction Models - The What, Why, How and It's Benefits. Express Analytics," 2022. [Online]. Available: https://www.expressanalytics.com/blog/everything-you-need-to-know-about-risk-prediction-models/.

[66] A. Ikhlaaq, T. Debray and K. Moons, "Developing and validating risk prediction models in an individual participant data meta-analysis," 2020. [Online]. Available: https://doi.org/10.1186/1471-2288-14-3.

[67] Y. Duan, J. Goodell, L. Haoran and L. Xinming, "Assessing machine learning for forecasting economic risk: Evidence from an expanded Chinese financial information set.," *Finance Research Letters,* vol. 46, no. A, 2021.

[68] D. Fung, "Methods for the estimation of missing values in time series.," 2023.

[69] "na_ma: Missing Value Imputation by Weighted Moving Average in imputeTS: Time Series Missing Value Imputation," [Online]. Available: https://rdrr.io/cran/imputeTS/man/na_ma.html. [Accessed 2023].

[70] NESTLER Consortium, D1.1 "NESTLER Platform Requirements", 2023.

[71] NESTLER Consortium, D3.1 "Remote Sensing Technologies and Multimodal Data Aggregation Protocols", 2023.