IoT-NGIN

D3.2
Enhancing Confidentiality Preserving Federated ML

WORKPACKAGE	WP3	**PROGRAMME IDENTIFIER**	H2020-ICT-2020-1
DOCUMENT	D3.2	**GRANT AGREEMENT ID**	957246
REVISION	V1.0	**START DATE OF THE PROJECT**	01/10/2020
DELIVERY DATE	30/11/2021	**DURATION**	3 YEARS

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

DISCLAIMER

ACKNOWLEDGEMENT

This document is a deliverable of IoT-NGIN project. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 957246.

The opinions expressed in this document reflect only the author's view and in no way reflect the European Commission's opinions. The European Commission is not responsible for any use that may be made of the information it contains.

PROJECT ACRONYM	IoT-NGIN
PROJECT TITLE	Next Generation IoT as part of Next Generation Internet
CALL ID	H2020-ICT-2020-1
CALL NAME	Information and Communication Technologies
TOPIC	ICT-56-2020 - Next Generation Internet of Things
TYPE OF ACTION	Research and Innovation Action
COORDINATOR	Capgemini Technology Services (CAP)
PRINCIPAL CONTRACTORS	Atos Spain S.A. (ATOS), ERICSSON GmbH (EDD), ABB Oy (ABB), INTRASOFT International S.A. (INTRA), Engineering-Ingegneria Informatica SPA (ENG), Bosch Sistemas de Frenado S.L.U. (BOSCH), ASM Terni SpA (ASM), Forum Virium Helsinki (FVH), Optimum Technologies Pilroforikis S.A. (OPT), eBOS Technologies Ltd (EBOS), Privanova SAS (PRI), Synelixis Solutions S.A. (SYN), CUMUCORE Oy (CMC), Emotion s.r.l. (EMOT), AALTO-Korkeakoulusaatio (AALTO), i2CAT Foundation (I2CAT), Rheinisch-Westfälische Technische Hochschule Aachen (RWTH), Sorbonne Université (SU)
WORKPACKAGE	WP3
DELIVERABLE TYPE	REPORT
DISSEMINATION LEVEL	PUBLIC
DELIVERABLE STATE	FINAL
CONTRACTUAL DATE OF DELIVERY	30/11/2021
ACTUAL DATE OF DELIVERY	15/12/2021
DOCUMENT TITLE	Enhancing Confidentiality Preserving Federated ML
AUTHOR(S)	S. Bourou (SYN), A. El Saer (SYN), A. Voulkidis (SYN), T. Velivassaki (SYN), D. Calvo Alonso (ATOS), D. Skias (INTRA), A. Zalonis
REVIEWER(S)	D. Calvo (ATOS), D. Skias (INTRA)
ABSTRACT	SEE EXECUTIVE SUMMARY
HISTORY	SEE DOCUMENT HISTORY
KEYWORDS	Federated Learning, Machine Learning, privacy-preserving, confidentiality

Document History

Version	Date	Contributor(s)	Description
V0.1	05/07/2021	SYN	Table of Contents
V0.2	21/07/2021	SYN, INTRA	FL definition, challenges
V0.3	06/09/2021	SYN, INTRA	FL frameworks, algorithms
V0.4	08/10/2021	SYN, INTRA	Privacy-preserving techniques
V0.5	21/10/2021	SYN, INTRA	Comparative analysis of the state of the art
V0.6	09/11/2021	SYN, INTRA, ATOS	Privacy-preserving Federated Learning as a Service
V0.7	30/11/2021	SYN, INTRA	Contribution to privacy preservation in Federated Learning within IoT-NGIN
V0.8	06/12/2021	SYN, INTRA, ATOS	Contribution to privacy preservation in Federated Learning within IoT-NGIN
V0.9	10/12/2021	SYN	Overall enhancements; document consolidation
V0.9.1	13/12/2021	SYN	Overall enhancements; Ready for peer review
V0.9.2	14/12/2021	ATOS	Peer review
V0.9.3	14/12/2021	INTRA	Peer review
V0.9.4	15/12/2021	SYN	Updates addressing peer review comments
V1.0	15/12/2021	SYN, CAP	Quality check; Final version

Table of Contents

Contents

List of Figures

List of Tables

List of Acronyms and Abbreviations

AI	Artificial Intelligence
BE-DHFL	Blockchain-Empowered Decentralized Horizontal Federated Learning
CPU	Central Processing Unit
DL	Deep Learning
DP	Differential Privacy
DP-SGD	Differentially Private Stochastic Gradient Descent
EEG	Electroencephalography
FATE	Federated AI Technology Enabler
FedAvg	Federated Averaging
FedPAQ	Federated Learning Periodic Averaging and Quantization
FedSGD	Federated Stochastic Gradient Descent
FL	Federated Learning
F-RCCE	Federated Reinforce Client Contribution Evaluation
FRL	Federated Region-Learning
FTL	Federated Transfer Learning
GAN	Generative Adversarial Network
GDP	Global Differential Privacy
GDPR	General Data Protection Regulation
GHL	Gazelle Homomorphic Layer
GNI	Gazelle Network Inference
GPU	Graphics Processing Unit
HE	Homomorphic Encryption
HHHFL	Hierarchical Heterogeneous Horizontal Federated Learning
IaaS	Infrastructure as a Service
IID	Independent, Identically Distributed
IoT	Internet of Things
IPS	Intrusion Prevention System

LL	Living Labs
LDP	Local Differential Privacy
MAD	Malicious Attack Detector
ML	Machine Learning
MLaaS	Machine Learning as a Service
MMD	Maximum Mean Discrepancy
MMVFL	Multi-participant Multi-class Vertical Federated Learning
MPC	Multiparty Computation
NSM	Network Security Monitoring
OISF	Open Information Security Foundation
OOV	Out-Of-Vocabulary
PaaS	Platform as a Service
PATE	Private Aggregation of Teacher Ensembles
PADDLE	PArallel Distributed Deep LEarning
PPFLaaS	Privacy-Preserving Federated Learning as a Service
PSU	Private Set Union
RNN	Recurrent Neural Network
SMC	Secure Multiparty Computation
SplitNNs	Split Neural Networks
STPV	Semi-honest Third Party
TFF	TensorFlow Federated
UAV	Unmanned Aerial Vehicle
WP	Work Package
ZKP	Zero-Knowledge Proof

Executive Summary

This document consists of the Deliverable "D3.2: Enhancing Confidentiality preserving federated ML" of the European H2020-ICT-2018-20 project "IoT-NGIN: Next Generation IoT as part of Next Generation Internet". D3.2 includes all the mandatory state-of-the-art Federated Learning tools, methods and algorithms along with the essential privacy-preserving characteristics. In addition, it presents the main achievements of IoT-NGIN towards the privacy-preserving Federated Learning tools. The keystones of this document are summarized as follows:

1. Extensive analysis of the current state-of-the-art regarding FL tools, methods, algorithms and frameworks;
2. Thorough investigation of open-source Federated Learning frameworks and libraries;
3. Extensive analysis on privacy-preserving techniques and specifically on Differential Privacy, Secure Multi-Party Computation and Homomorphic Encryption;
4. Definition of the Privacy-Preserving Federated Learning as a Service (PPFLaaS) concept;
5. Presentation of three Federated Learning frameworks that are well suited for the IoT-NGIN scope, namely Flower, PySyft together with PyGrid and TensorFlow Federated;
6. Design of the privacy-preserving FL framework architecture, integrating Private Aggregation of Teacher Ensembles (PATE) technique with Flower Federated Learning framework, for the IoT-NGIN project;
7. Presentation of the developments towards the first version of the IoT-NGIN Privacy-Preserving Federated Learning framework, including porting of PATE in Keras and experimentation with the Flower Federated Learning framework.

The scope of this deliverable is to analyse the different Federated Learning techniques and methodologies through the lens of privacy-preserving and focus on the ML modelling to propose a federated ML module that will provide the IoT-NGIN ML framework with the required confidentiality-preserving features.

1 Introduction

Artificial Intelligence (AI) plays a growing role in Internet of Things (IoT) applications. One can say that the main added value of AI is its ability to produce insights, to automatically identify patterns and to detect anomalies on data collected or generated using IoT sensors and other devices. Machine Learning (ML), as a specific subset of AI that trains machines on how to learn from data, has efficiently contributed to many aspects, both for businesses and consumers, including proactive intervention, tailored experiences and intelligent automation. ML is almost everywhere nowadays, from small wearable devices and smartphones to powerful super-computers ensuring fast and accurate data analysis. Moreover, IoT devices generate a great amount of data every day and thus, raise significant concerns about privacy and ownership of the collected or generated data.

Traditional cloud computing applications need the data to be uploaded and processed on a central server giving data access to third parties. This raises privacy and data ownership concerns. Furthermore, IoT devices are already capable of processing a vast amount of data due to their powerful hardware specifications making it possible for local data processing and analysis. Thus, edge computing is witnessing great interest especially after the emergence of 5G.

Nevertheless, data privacy is the most fundamental objective regarding data access and processing. This has led to the elaboration of strict data privacy legislations such as the Consumer Privacy Bill of Rights in the U.S. and the European Commission's General Data Protection Regulation (GDPR). For example, Articles 5 and 6 of the GDPR state that data collection and storage should be restricted to only what is user-consented and decidedly indispensable for processing.

To address privacy issues, Google [1] introduced *Federated Learning (FL)*, a specific approach in edge computing. Federated Learning is able to overcome the privacy concerns that emerge in a central cloud-based architecture by enabling an *on-device* collaborative training of a machine learning model without sharing any data over the network. This is achieved by initializing the training of a global machine learning model on a central server for a few iterations to obtain some initial weights. These model weights are then sent to the participants (data owners), which use their own resources to locally train the machine learning model. After training, each client sends its own updated weights to the server, which is responsible to aggregate the weights from all the different clients and produce a new global model. This process is repeated for several iterations until the global model reaches a certain desired accuracy level or reaches the limit number of iterations. Federated Learning aims to train an ML privately by sharing model parameters (weights of the model) than sharing the data itself. This feature enables machine learning models to run on local and private data. However, model sharing can also potentially reveal sensitive information. Therefore, FL needs additional privacy-preserving techniques to enable fully private machine learning model sharing and training. Differential Privacy (DP), Secure Multiparty Computation and Homomorphic Encryption (HE) constitute the most popular privacy-preserving techniques for FL systems.

Considering the above, IoT-NGIN provides Federated Learning frameworks with enhanced privacy-preserving characteristics for the *Machine Learning as a Service (MLaaS)* platform. Specifically, one of IoT-NGINs' main objectives is to design and develop privacy-preserving

FL frameworks which shall include certain methodologies to ensure confidential data isolation.

The present document entitled "D3.2: Enhancing Confidentiality preserving federated ML" is a technical report which includes the state-of-the-art regarding Federated Learning with enhanced privacy-preserving techniques. Additionally, it presents the Federated Learning tools and methods that are designed and developed during the IoT-NGIN project lifecycle.

1.1 Intended Audience

The intended audience includes ML engineers and developers which may find this report intuitive for their research efforts as well as ML service providers who aim to enhance their services with privacy-preserving techniques. This deliverable focuses to the utmost extent, on privacy-preserving Federated Learning and its usability through the inter-connected tools in the IoT-NGINs' MLaaS platform. More specifically, this document provides an extensive analysis of the state-of-art FL tools, algorithms, privacy-preserving techniques and application scenarios. Thus, anyone who wants to engage with FL can take advantage of this document.

Moreover, IoT-NGINs' vision for edge computing through Federated Learning methods could potentially define new methodologies, services and tools, which enhance already implemented solutions and raise awareness about privacy.

Finally, this report is useful internally, to the project partners and especially Work Package (WP) 3 and WP5. Useful feedback could be also received from the Advisory Board, including both technical and impact creation comments.

1.2 Relations to other activities

This deliverable primarily relates to Task 3.1 as the privacy-preserving FL frameworks which task 3.3 will provide, will feed future activities towards building the Machine Learning as a Service (MLaaS) platform of the IoT-NGIN project. Additionally, the FL components will be included in the Digital twin System which is defined in Task 5.5. The tools which are described in this document will support the deployment of Task 5.2 and especially, shall be able to easily integrate the tool from Task 5.2, namely *MAD* (Malicious Attack Detector). Furthermore, the tools described here will provide assistance -if needed- to the *Open Calls*. Last, but not least, the document provides useful feedback to the future integration and validation activities, as well as to the preparation of the trials in the Living Labs (LLs) in WP7.

1.3 Document overview

The rest of the document is divided into the following sections:

- Section 2 provides an extended literature review of Federated Learning, presenting teh definition and challenges of the FL approach, as well as analyzing state-of-the-art techniques, frameworks and algorithms.
- Section 3 addresses privacy preservation in Federated Learning, presenting state-of-the-art techniques and conducting a comprehensive comparative analysis of them.
- Section 4 presents the IoT-NGIN approach towards enhancing FL with privacy-preserving techniques, introducing the Privacy-Preserving FL as a Service of IoT-NGIN,

the privacy-preserving FL frameworks that will be supported by IoT-NGIN, as well as the privacy-preserving enhancements over existing FL frameworks introduced by the project.

- Section 5 draws conclusions and next steps.

2 Federated Learning Background

FL as a new breed of Artificial Intelligence aims to build upon decentralized data and train machine learning models on the edge while avoiding data disclosure. Thus, as opposed to traditional approaches, FL intrinsically enhances privacy and security as the data is never accessed or processed on central servers. Thus, decoupling the training process from the data sources. FL approaches are preferred in cases where security and privacy are the key concerns or when the participants are not always available, or their network connectivity is unstable.

Conventional machine learning methods demand the data for training to be located on one machine or on a central server. Those methods collect a great amount of data from various devices (smartphones, laptops, IoT devices) and transfer them to a high-end central server or a super strong computer, for training. However, data owners are not often willing to share their data, because it is sensitive information, which is subject to the GDPR rules. Thus, applying ML models on data without revealing the data itself is a major task, which many researchers have tried to address during recent years. In particular, Google [2]–[5] introduced Federated Learning, which extends further, traditional distributed machine learning approaches. Actually, FL emerged as an extended subset of distributed learning. The main difference between Federated Learning and traditional distributed learning lies in the fact that distributed learning aims to parallelize computing resources to train a machine learning model on centralized data, whereas Federated Learning aims at training a shared machine learning model on local datasets.

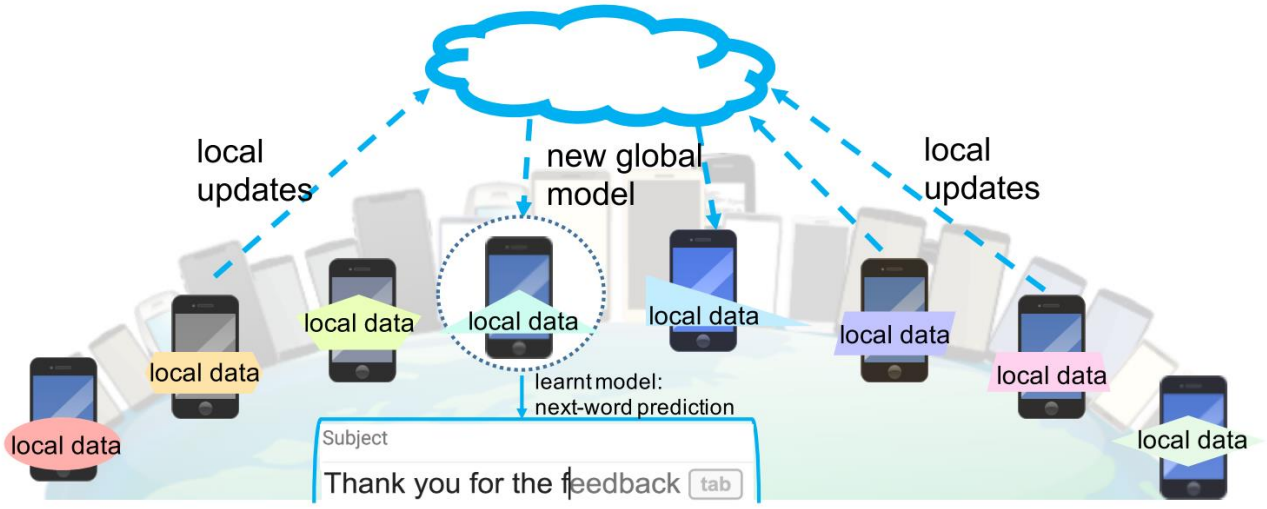


Figure 1: A federated application for *next-word* prediction on mobile phones. Local data remain local during training to preserve privacy. Each mobile device may communicate periodically or for a short time with the central server [6].

Initially, FL solutions train a machine learning model on a central server. Afterwards, the server shares this model to the participating devices for local training, thus without revealing any data. Then, the updated model parameters are sent back to the central server and the global model is updated by averaging the results from each participant. This task is repeated until the final model is produced. Figure 1 presents a federated learning application which trains a machine learning model to predict the next word while the user is typing. FL methods

are capable of training robust and resilient machine learning models based on model sharing rather than data sharing over a network.

FL approaches can be applied in several application scenarios, where data privacy is essential, listing from cross-organizational scenarios such as health care institutes to cross-devices like smartphones, wearable and IoT devices with millions of participants. Due to the growing computational capabilities of smartphones, wearable and IoT devices, it is increasingly attractive to store data locally and perform machine learning on the edge. Some other examples of Federated Learning applications include sentiment learning and analysis, predicting health events from wearable devices and so on.

Nevertheless, federated learning poses some limitations. To begin with, some machine learning models are quite large to run on low-end devices. Furthermore, there can be great data heterogeneity, meaning that data among users may be not relevant or of different characteristics. Normally, before applying machine learning models on datasets, data pre-processing is needed which includes data cleaning, data augmentation (in case of limited available data), removing misleading data entries. In the case of Federated Learning, data preprocessing is not applicable since it is manual work but for sure, some pre-process automation may be feasible. Most machine learning models are supervised, meaning that data labelling is essential. This can be a drawback for some applications and thus, federated learning is better suited for unsupervised learning applications.

2.1 Federated Learning Challenges

Federated Learning is an active and ongoing area of research. In recent years many surveys and reviews listed almost every challenge in Federated Learning systems [7], [8], [9]. In this chapter, an extended analysis is given which accompanies two tables. Table 1 lists the main known challenges so the reader can use it as a relevant dictionary. The first challenge relates to Communication which can be a bottleneck for FL systems that includes hundreds or millions of devices. Another core challenge that FL developers face Is the system heterogeneity of the devices since their hardware resources can be significantly different to each other and also their connection capabilities as well. Usually, the participated devices may drop out during training because of low battery or other difficulties that may emerge. A data-related challenge is the statistical heterogeneity issue that typically exists among participants. The main reason comes from the fact that devices may preserve locally or collect, data of different distributions. Finally, privacy and security are really important challenges to tackle and a strong open research field in Federated Learning.

Table 1: Summary of main challenges in Federated Learning systems.

Challenge Category	Challenge	Main Idea
Communication	Model Updates Size Reduction – Compression Schemes	Reduce the size of the model updates that are exchanged between the clients and the server by compressing them

	Communication Frequency Reduction	Decrease the communication frequency by reducing the number of parameters or performing aggregation periodically or over-the-air computation meaning that whole or parts of the aggregation are carried out over-the-air by clients
	Change Communication Type	The server and the clients can communicate in a synchronous or asynchronous way
Systems Heterogeneity	Resource Management	How to efficiently manage the limited resources on client devices to maximize performance
	Client Management	Increasing the number of clients that participate during the training is a critical step towards maximizing performance
	Fault Tolerance	Fault tolerance is the problem of client dropout. A dropout that may occur at any moment and under any circumstances
	Client Selection Based on Reliability	In cross-device applications, there can be millions of clients where some of them may be unreliable
Statistical Heterogeneity	Non-IID Data	This challenge is because of the inherent heterogeneity in the local data
	Model Heterogeneity	Model heterogeneity results when a participant designs its own training model
	Bias Mitigation	The presence of Bias in Machine Learning models may lead to discrimination and unpleasant situations
Privacy/Security	Byzantine attacks	Attacks that aim to deteriorate the model's performance or cause a failure during training
	Privacy-preserving	Privacy-preserving techniques such as Differential Privacy, Homomorphic Encryption and Multi-party Computation should be employed in Federated Learning systems

	Secure Aggregation	Refers to the challenge of computing the sum of the local gradients updates from many participants without revealing information

Table 2: Summary of some major approaches that address Federated learning challenges.

Approach	Challenge Category	Core Idea
[10]	**Communication**	An online Federated distillation where each device sees the mean model output of all other devices and periodically measures the difference using cross-entropy that becomes the student's loss legularizer, obtaining the knowledge of the other devices during the distributed training process
[11]	**Communication**	To decrease communication costs. Clients quantize their locally computed gradients and send the result to the central server
[12]	**Communication**	Introduce the Sparse Ternary Compression, a framework which relies on top-k gradient sparsification with a novel mechanism to enable downstream and also, ternarization and optimal Golomb encoding of the weight updates
[13]	**Communication**	In-Edge AI framework is proposed to enable collaborative devices exchanging learning parameters in a more sophisticated way leading to communication reduction
[14]	**Systems Heterogeneity**	Authors propose a double Deep Q-Network (DQN) approach that assists the model owner to i) decide amounts of energy recharged to the workers and ii) choose channels, i.e., the default channel or the special channels, for the global model transmissions to maximize the number of successful transmissions while minimizing the energy cost and the channel cost

[15]	**Systems Heterogeneity**	Present an algorithm to determine and analyse, the frequency of global aggregation so that the available resource is most efficiently used
[16]	**Systems Heterogeneity**	Authors propose SAFA, a semi-asynchronous FL protocol, to address the problems in federated learning such as low round efficiency and poor convergence rate in extreme conditions, e.g., clients dropping offline frequently
[17]	**Systems Heterogeneity**	A Deep Q-Learning approach that enables the server to learn and find optimal decisions without any a priori knowledge on client devices
[18]	**Statistical Heterogeneity**	This approach proposes FedMD, a federated learning framework that enables participants to independently design their models. The key element of FedMD is that it translates knowledge between participants
[19]	**Statistical Heterogeneity**	Presents FAVOR, an experience-driven control framework that chooses the clients to participate in each round of federated learning to counterbalance the bias introduced by non-IID data and to speed up convergence. More specifically, the mechanism is based on a Deep Q-learning method that learns to select a subset of devices in each communication round to maximize a reward that encourages the increase of validation accuracy and penalizes the use of more communication rounds
[20]	**Statistical Heterogeneity**	This approach proposes the use of a new federated aggregation scheme that converges even when devices may be inactive or return incomplete updates while relaxing the restrictions regarding client participation in FL settings and allowing devices to follow more flexible participation patterns

[21]	**Statistical Heterogeneity**	In this work, an algorithm for the adaptation of the learning rate for deep learning stochastic gradient descent that avoids the need for validation set use, is proposed
[22]	**Privacy/Security**	To increase attack stealth, the authors propose an alternating minimization strategy, which alternately optimizes for the training loss and the adversarial objective
[23]	**Privacy/Security**	The paper presents a Byzantine resilience method that forces the vector output selected by the server to point, on average, to the same direction as the gradient
[24]	**Privacy/Security**	Authors present a collaborative and privacy-preserving machine teaching paradigm with multiple distributed teachers, to improve robustness of the federated training process against local data corruption
[25]	**Privacy/Security**	Clients train a general model on the task and then adapt this general model to each user's private domain while simultaneously, learns a private model for each user. Then, each user combines these models using a mixture of experts (MoE) [26]

The most important drawback until now has been the hardness of reducing communication overheads during training. To address the huge amount of data available for training and make Federated Learning more generic, researchers have made effective efforts towards improving model updates during training, reducing communication costs for each training round and developing finer communication schemes.

In network theory, two types of communications exist in a server and client network, the downstream and the upstream, where downstream communication refers to the model sharing from the server to each client and upstream towards the opposite direction. The authors in [27] present the *FederatedAveraging* algorithm which is based on iterative model averaging which reduces heavily the communication rounds. In [28] the authors propose *FedCS*, a novel Federated Learning protocol that tackles the difficulties which emerge, when mobile devices of different data resources, wireless channel conditions and computational capabilities participate in a Federated Learning setup. *FedCS* framework integrates the available clients to the maximum extent in each training round by managing them based on their resource conditions. The approach in [29] utilizes a two-stream model with Maximum

Mean Discrepancy (MMD) [30] constraint during each training iteration which forces the local two-stream model to accumulate knowledge from other devices. This method brings a communication cost reduction without affecting the final performance of the training.

2.2 Federated Learning vs Distributed Learning

Federated Learning is a step beyond the traditional distributed machine learning (Figure 2) that exploits data parallelism. Rather than portioning a centralized dataset to random nodes, Federated Learning performs the training in the data owned by each participant (FL clients), as shown in Figure 3. The main added value is that the data itself is never shared over the network. Federated Learning is far more distributed than conventional distributed machine learning.

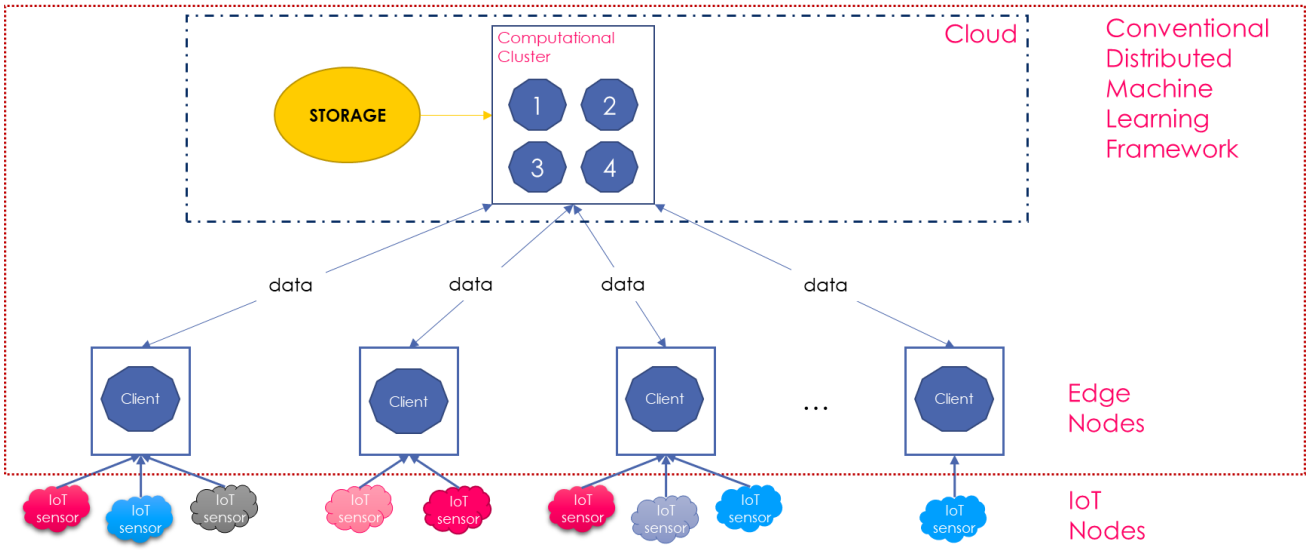


Figure 2: Traditional distributed machine learning. There is a central Computational cluster, and the architecture is based on data exchange.

The authors in [26] list the core differences between distributed machine learning and federated learning as follows:

- Contribution from many clients: Federated Learning should support a large pool of participants and must be scalable.
- Varying quantity of data owned by each participant: Some clients may have a few samples for training and others may have thousands.
- Common differences in data distributions among participants: Local data is different for each participant; thus, the model trained by each client represents non-IID (independent, identically distributed) data.
- High communication latency between participants and the aggregator server: Updates are commonly shared over the network introducing significant latency between communication rounds.
- Unstable communication between clients and aggregating service - client devices are likely to become unavailable during training due to their mobility, battery

Unlike the traditional distributed learning systems, FL systems do not preserve control over participants' devices.

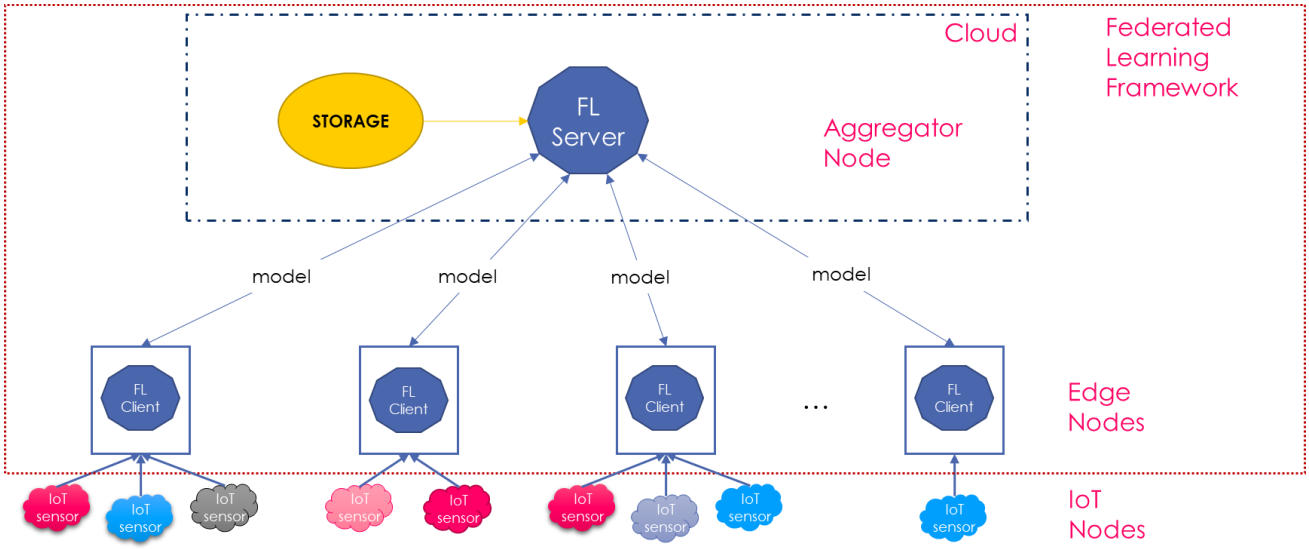


Figure 3: Federated Learning. The model is shared, and the training is performed on the edge nodes.

Federated Learning biggest advantages over conventional distributed machine learning techniques are listed below, so FL:

- Decreases by far, the risk of unauthorized data access, since data are not transmitted over the network.
- Takes advantage of privacy-preserving techniques as they can more easily be integrated into the processing chain compared to distributed learning.
- Needs fewer resources than conventional distributed machine learning.
- Enables multiple participants to build a robust and resilient machine learning model that on the one hand and on the other does not share data, thus allowing them to face critical issues in terms of privacy and security.
- Can be applied to heterogeneous datasets of different sizes.
- Can be asynchronous and does not rely only on powerful devices.

2.3 A Categorization of Federated Learning

Federated Learning is a model-centric task which trains collaboratively, a shared model on decentralized data. Each client preserves data locally and private, meaning that clients cannot access or alter the data of other clients. Federated Learning can be divided into two main categories, Cross-device, and Cross-silo. Moreover, federated learning may be classified into horizontally federated learning, vertically federated learning and federated transfer learning based on how data is distributed among various parties in the feature and sample space [32].

2.3.1 Cross-Silo Federated Learning

In Cross-Silo [31], [32] Federated Learning, clients can be different organizations in various domains such as healthcare and financial. Data is held locally and remains decentralized throughout the learning procedure. In each training round all clients do participate in the

learning process compared to Cross-device Federated Learning, where typically, a fraction of clients participates in each round. In addition, each client stores its own independent data locally. In Cross-Silo systems there is a limited number of clients, typically 2-100 and each client holds a unique identity or name that allow the system to access it specifically. The main issue in Cross-Silo Federated Learning settings might be computation or communication specific.

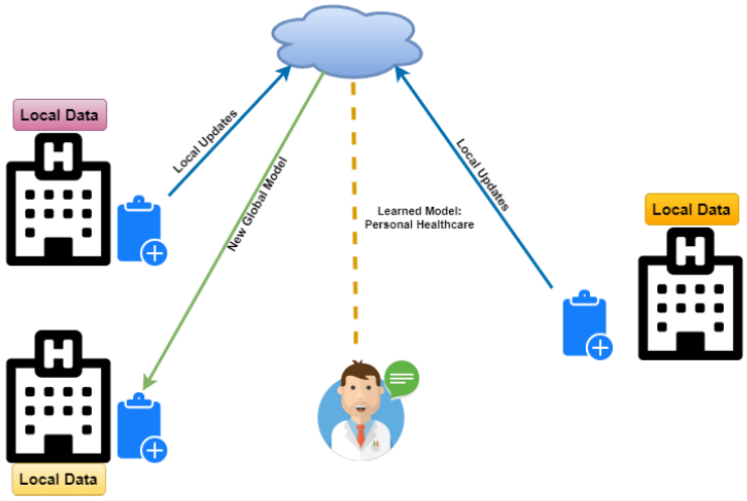


Figure 4: Cross-Silo Federated Learning in Healthcare domain, [33].

A Cross-Silo Federated Learning architecture example is presented in Figure 4. Cross-Silo Federated Learning enables the use of more consistent, powerful, and scalable tools due to the nature of the participants which typically have quite enough resources.

2.3.2 Cross-Device Federated Learning

A Cross-Device [4], [26], [32] Federated Learning system typically includes many clients, sometimes even millions. Similar to Cross-Silo, Cross-Device Federated Learning is applied on decentralized local data and in each training round, the central server aggregates the updated model parameters to compute the global model. Communication is often the primary issue because Cross-Device Federated Learning uses Wi-Fi or slower connection protocols. Another challenge to address is that Cross-Device is often unreliable because many clients fail or dropout (battery issue, network connectivity, etc.) and therefore may not be able to contribute to the learning process.

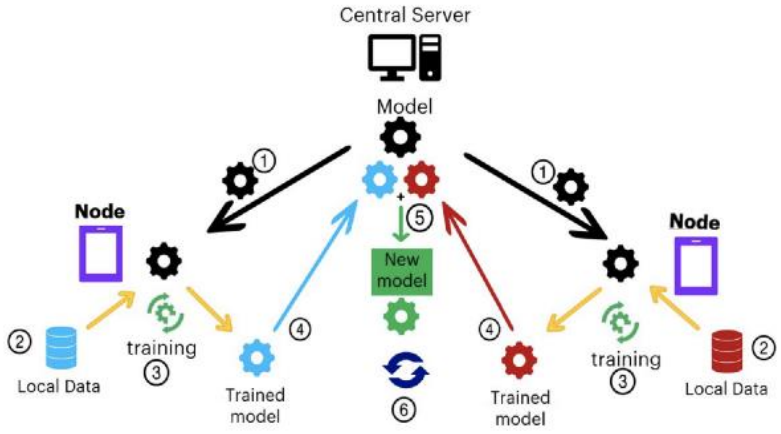


Figure 5: Architecture overview of cross-device Federated Learning, [34].

In Figure 5, a basic Architecture overview for Cross-Device Federated Learning is presented. In Cross-Device Federated Learning, the number of training rounds can vary from one device to another while the final model may be deployed to thousands, or even more clients and devices may have either white-box or black-box access to the final model. Therefore, it is important to preserve privacy in between the training rounds and especially from the intermediate training state until the final model. Participant devices need to trust the central server for privacy-preserving during the training but also must trust the server to form cohorts of clients.

2.3.3 Horizontal Federated Learning

In conventional machine learning scenarios, the data and the training are all done in a centralized manner while the data is often obtained from data centres. Horizontal Federated Learning (Figure 6) can be related up to a point, with distributed machine learning. The main difference is that the data used in Federated Learning is local instead of owned by the central server. The basic steps in a Horizontal Federated Learning scenario are as follows:

- Participants download the latest model from server A.
- Each participant trains the model independently on their own local data and uploads the model updates to the central server, once the training is done.
- Server A returns the model to every participant.
- And finally, each participant updates its own model.

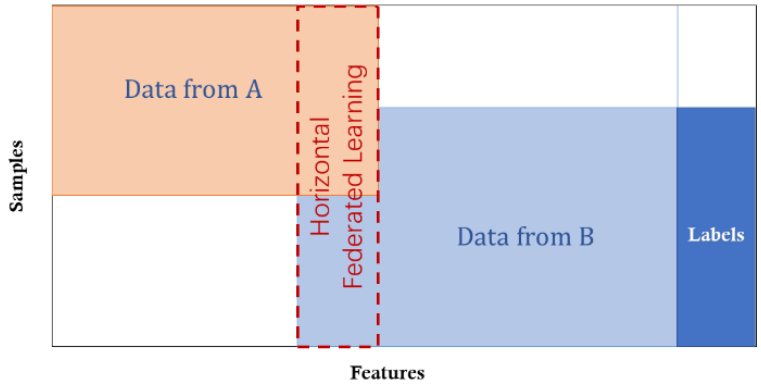


Figure 6: Horizontal Federated Learning, [32].

In Horizontal Federated Learning, datasets share the same feature space but consist of different samples. This means, that datasets share the same feature description (e.g., age, name, height, etc.) but preserve different sample entities. Horizontal Federated Learning has been employed by many researchers and in various use case scenarios. In [35], the authors propose a Hierarchical Heterogeneous Horizontal Federated Learning (HHHFL) approach to train a machine learning model over heterogeneous Electroencephalography (EEG) data, while preserving data privacy for each participant. In [36], the presented approach employs horizontal Federated Learning for training a reinforcement learning model and introduces the Federated REINFORCE Client Contribution Evaluation (F-RCCE). Authors in [37], present a Blockchain-Empowered Decentralized Horizontal Federated Learning (BE-DHFL) framework for 5G-enabled Unmanned Aerial Vehicles (UAVs).

2.3.4 Vertical Federated Learning

Vertical Federated Learning or Heterogeneous Federated Learning is applicable to the cases where two datasets share the same identities but preserve different sets of features. Vertical Federated Learning (Figure 7) allows multiple parties that own different attributes of the same data entities to jointly train a machine learning model. For example, vertical Federated Learning can be used between insurers and online retailers where both of them have lots of overlapped users, but each one preserves its own feature space and labels.

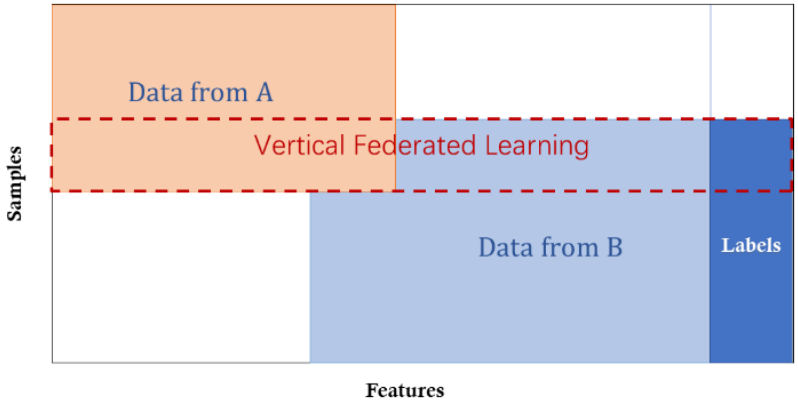


Figure 7: Vertical Federated Learning, [32].

Vertical Federated Learning is often used when there is an increased user space overlap in datasets among different organizations. Moreover, vertical Federated Learning allows the aggregation of the split features of shared entities without interfering with the privacy preservation requirements. Many machine learning models are built in a vertical Federated Learning way. In [38], the Pivot tool is proposed for privacy-preserving vertical decision tree training and prediction. Pivot ensures information is private except when participants want to release their data. Authors in [39] present a quasi-Newton method for vertical Federated Learning that reduces communication cost and is used for logistic regression using additively homomorphic encryption to meet also the necessary privacy requirements. Other approaches [40], [41], use Split Neural Networks (SplitNNs) to perform machine learning on vertically distributed datasets in a two-party scheme. More specific, SplitNNs map the data into an abstract, shareable representation which allows information from multiple sources to be combined for learning without exposing raw data. Most proposed methods for Vertical federated learning are built upon a two-party setup. To overcome this limitation, authors in [42], proposed the Multi-participant Multi-class Vertical Federated Learning (MMVFL) framework for multi-class VFL scenarios which includes multiple parties. MMVFL learns a separate model for each participant, instead of a single global model to enable personalized learning. To enhance privacy-preserving, authors in [43], propose a vertical Federated Learning framework based on Private Set Union (PSU), a protocol that can securely align data entities and allows every party to keep sensitive information private while training the model. A vertical Federated Learning system typically assumes honest-but-curious clients. To facilitate secure computations between two parties in a Vertical Federated Learning setup, sometimes a Semi-honest Third Party (STPV) is introduced. By the end of the learning procedure, each party holds only the relevant model parameters, but this means that at inference time, these two parties must collaborate to generate the output. In [45], a vertical Federated Learning architecture for logistic regression that removes the role of the third-party coordinator is presented which reduces the complexity of the system and enables faster and more private training.

2.3.5 Federated Transfer Learning

Federated Transfer Learning (FTL) was introduced in [45] to address the scattered nature of the data across different parties which cannot easily be involved in a learning process due to privacy or other limitations. A simple architecture overview is presented in Figure 8. FTL enables knowledge sharing without revealing sensitive information. Federated Transfer learning aims to build an effective model for a use case scenario that holds limited data, or data with limited labelling or the resources are small. FTL can be utilized in various scenarios. For example, in [46], the authors present FedHealth, a Federated transfer learning framework for wearable healthcare to tackle data privacy and data availability. FTL can bridge the gap between machine learning and information privacy to address the limitations presented above.

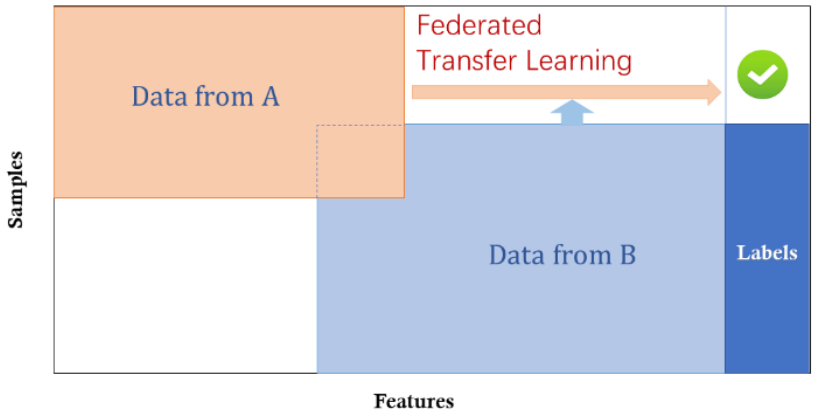


Figure 8. Federated Transfer Learning, [31]

Authors in [47], propose a deep neural network to train on multiple devices where the data input are covariance matrices of electroencephalographic (EEG) signals for adaptive analysis.

2.4 Applications in Federated Learning

Several applications have been proposed that employ FL in various domains such as mobile phones, IoT devices, industry, and healthcare. Some approaches improve predictions on keyboards or emojis within an FL setting environment exploiting ML models that take advantage of a small amount of user-generated preceding text. The application in [48] learns a predictor in a large-scale smartphone network based on users' text data. In [49] the authors propose a character-level Recurrent Neural Network (RNN) to learn out-of-vocabulary (OOV) words to expand the vocabulary of a virtual keyboard for mobile phones without exporting the sensitive text to servers. Other approaches such as [50] use FL to solve out-of-domain issues with continuously running embedded speech-based models such as wake word detectors for voice assistants. Yang et. al. in [51] use FL in a global-scale setting to train, evaluate and deploy an ML model to improve virtual keyboard search suggestion quality avoiding any data disclosure. In [52] a word-level RNN is used to predict emojis by pretraining the model using a language modelling task.

With respect to federated learning in IoT environments, there are three main challenges to consider, device, statistical and model heterogeneity. Device heterogeneity refers to different usage environments and different resources of each IoT participant. That includes differences in hardware (CPU, GPU), network conditions and battery availability. Regarding statistical heterogeneity, there are big differences in user patterns and the environmental characteristics of each participant. Thus, in IoT environments data distributions are highly skewed, which may lead to nondivergence during training. Model heterogeneity refers to the fact that not every model architecture can fit on every IoT device. As a consequence, the model architectures from different participants are of various shapes making it really difficult to perform aggregation methods in FL settings. Therefore, recent FL studies have shown that there is a need for personalized federated learning. Some FL applications, cope with these difficulties in IoT networks and try to learn users' preferences in smart home environments to address privacy leakage due to cyber threats or physical hazards. Matchi et. al in [53] propose a federated learning framework to train machine learning models locally, combined with a secure data aggregation protocol. In [54] the authors present a

federated multi-task learning framework to automatically learn customized context-aware control policies from the observed behavioural patterns of users in smart home environments in a privacy-preserving manner. Chen et al. [46] presented a method which focuses on personalized training. Initially, the proposed method trains a global ML model in a federated manner and then transfer it to each device for further personalized training. This process creates a training overhead and that is why the authors fine-tune only parameters of specified layers of the network instead of retraining the whole model. Similar in [55], the proposed framework, applies federated transfer learning and divides the deep learning models into two subcategories, the base, and the personalization layers. Were base layers act as shared layers which are trained using the traditional federated learning method. In contrast, the personalization layers are trained locally thus, personal information is strongly accounted during training while privacy is preserved.

FL is quite strong in data privacy protection and has emerged as a promising solution in various industrial applications. Industry 4.0 enhances the operational efficiency of the entire manufacturing process by integrating multiple disruptive technologies such as the Internet of Things, Edge Computing and Artificial Intelligence. For example, authors in [56], proposed an inference Federated framework namely Federated Region-Learning (FRL) for urban sensing. FRL considers the regional specifications during the distribution of training samples to improve inference accuracy. Federated Learning is also applied in Flying Ad-hoc Networking [57]. A federated on-device learning for jamming attack detection security architecture for FANET.

Federated Learning is suitable also for Healthcare applications. For instance, hospitals are organizations that private data of patients, that should remain local. FL architecture presented in [2] can reduce privacy leakage and implement private learning between the different organizations. Additionally, FL can also be used on IoT networks to ensure privacy, enabling on-device machine learning solutions without the need to store private data from end devices to a central server.

Nowadays, to achieve good results during ML model training, a great amount of data is needed. In addition to this, IoT devices can perform a series of tasks due to powerful hardware specifications, breaking the computational barrier of the previous years. All these advancements in IoT devices raise a privacy question. Malicious adversaries can compromise the privacy of IoT devices by tracing the vulnerabilities to manipulate ML model outputs or to get access to sensitive data. To an extent, federated learning solves privacy concerns however, additional security measures are essential.

2.5 Federated Learning Frameworks

Several open-source Federated Learning frameworks have been developed to apply distributed learning on decentralized data but also to enhance privacy and security. Google proposed *TensorFlow Federated* [58] an open-source framework for machine learning and other computations on decentralized data. Another open-source federated learning framework is *PySyft*, which was introduced by OpenMined [59]. *PySyft* is suitable for research in FL and allows the users to perform private and secure Deep learning. *PySyft* is also integrated into *PyGrid* [4], a peer-to-peer platform for federated learning and data privacy, which can be used for private statistical analysis on the private dataset as well as for performing FL across multiple organization's datasets. WeBank's AI department introduced *FATE* (Federated AI Technology Enabler) [60] an open-source framework which supports FL

architectures and secure computation of various machine learning algorithms. *FATE* is an industrial-grade framework mostly oriented toward enterprise solutions. Authors in [61] presented *Flower*, a friendly open-source federated learning framework which is ML framework agnostic and provides higher-level abstractions to enable researchers to experiment and implement on top of a reliable stack. Another promising open-source Federated Learning framework is *Sherpa.ai* which is presented in [62] and incorporates federated learning with differential privacy. *Sherpa.ai* results as a combination of machine learning applications in a federated manner with differential privacy guidelines. *FedML* [63] is an open-source federated learning 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. FedML promotes diverse algorithmic research due to the generic API design and the comprehensive reference baseline implementations. Another well-known open-source federated learning framework is the *PaddleFL* [64]. In *PaddleFL* researchers can easily replicate and compare different federated learning algorithms while they can easily be deployed in large scale scenarios. *Leaf* [65] is a modular benchmarking framework for federated learning with applications including Federated Learning, multi-task learning, meta-learning, and on-device learning. *OpenFL* [66], is another open-source federated learning framework for training ML algorithms using the data-private collaborative learning paradigm of FL. *OpenFL* works with machine learning pipelines built on top of TensorFlow and PyTorch and is easily customizable to support other machine learning and deep learning frameworks. In the following sub-sections, a more extended analysis is given for each framework. In section 3.2, a thorough comparative analysis on these federated learning frameworks is presented towards the scope of IoT-NGIN.

2.5.1.1 TensorFlow Federated (TFF)

TFF [58] is an extensible, federated learning framework for conducting federated learning research by simulating federated computations on realistic proxy datasets. TensorFlow federated includes *TensorFlow privacy*, a python library for applying privacy techniques in machine learning model training. Additionally, there are many helpful tutorials for image classification and text analysis while it is easily integrated with existing *TensorFlow* machine learning models. *TFF* provides two different APIs, the federated learning, and the federated core. The Federated Learning API offers a set of high-level interfaces while the federated core API offers a set of low-level interfaces for customized federated algorithms implementation, performing computations that involve multiple participants. In Figure 9, an architecture overview of *TFF* is presented. *TFF* implements *FedAvg* and Federated Stochastic Gradient Descent (*FedSGD*) algorithms. *FedAvg* in *TFF* uses three main aggregate operations:

- *Sum* – which sums clients' values and publishes the result at the central server
- *Mean* – which computes the weighted mean of clients' values and publishes the result at the central server
- *Differentially private* – which computes a Gaussian or a Laplacian noise base on the values, then add this noise to the data and publishes the result at the central server

Regarding the drawbacks of *TFF*, to begin with, it doesn't support federated mode, meaning that it simply cannot run a *real-life* experiment or for commercial reasons. Another issue is that vertical and hybrid data splitting is not supported. Regarding privacy-preserving mechanisms, TFF privacy includes *Private Aggregation of Teacher Ensembles* (*PATE*) which

will is thoroughly presented in section 3.1.4. Yet, *TFF* is supported by a big community with many contributors.

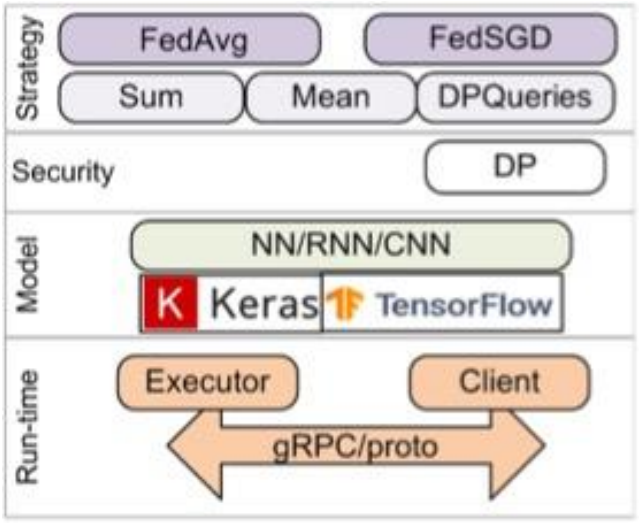


Figure 9: Architecture Overview of TensorFlow Federated (TFF), [67].

2.5.1.2 PySyft + PyGrid

PySyft is an open-source federated learning tool for secure deep learning with an MIT license. PySyft decouples private data from model training by applying differential privacy and Secure multi-party (MPC) mechanisms. It is one of the most prominent federated learning frameworks since it has the biggest support community. PySyft is more of a simulation than an actual production-ready solution therefore is well-suited for research purposes and prototyping. Some of the main features of *PySyft* are the following:

- It is compatible with existing Deep Learning frameworks such as TensorFlow and PyTorch.
- *PySyft* provides a low-level Federated Learning implementation that facilitates the development and debugging of the projects
- *PySyft* privacy mechanisms focus on secure - MPC through Homomorphic encryption, therefore, enabling computations on ciphertexts, which is a great advantage for developing Federated Learning applications

Along with PySyft, in the OpenMined ecosystem the following projects are included:

- PyGrid – A peer to peer network of data owners and data scientists who can collectively train models using PySyft
- KotlinSyft – Is an Android-based tool that gives the opportunity to train and apply inference of PySyft models
- SwiftSyft – It is KotlinSyfts' correspondence for iOS devices
- Syft.js – It is a web interface component for the above-mentioned components

The main shortcoming of PySyft is that it functions properly only in simulation mode and to support federated mode it needs to be incorporated with PyGrid.

2.5.1.3 FATE

FATE (Federated AI Technology Enabler) [60], is an open-source project initiated by WeBank's AI Department and intended to provide a secure computing framework to support federated machine learning applications mainly for regression analysis and decision trees while it also provides transfer learning. *FATE* is designed for industrial applications and some major characteristics are listed as follows:

- The framework includes frequently used horizontal and vertical federated algorithms for data engineering and machine learning. Additionally, provides a workflow engine to develop customizable full-lifecycle machine learning tasks
- Provides a self-developed distributed computing, transmission, and storage engine for large-scale applications.
- Exposes a high-level interface driven by custom scripts and supports many FL algorithms.
- To ensure security, *FATE* supports Multi-party Computation mechanisms and encryption methods such as RSA, Paillier and homomorphic encryption.
- *FATE* supports simulation and federated mode using Kubernetes clusterization.

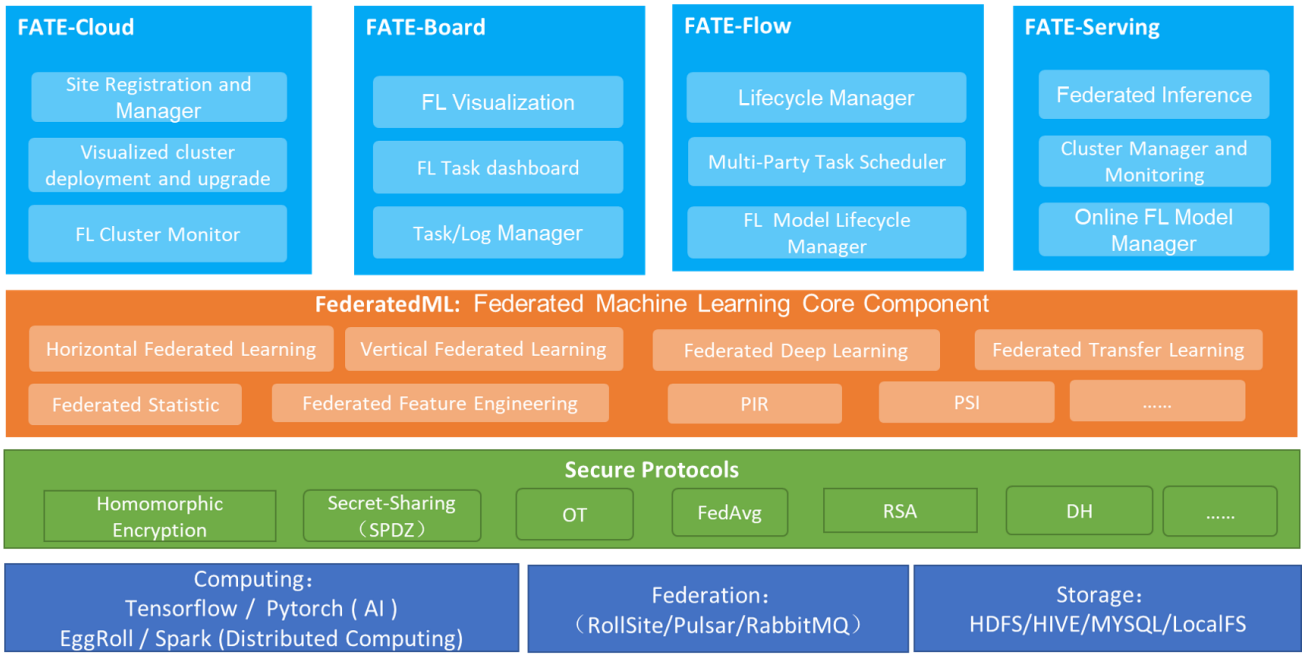


Figure 10: Architecture overview of FATE framework [60].

In Figure 10, an overview of FATE's architecture is presented. Following, the four independent platforms are listed:

- *FATE-Cloud* includes a cloud manager which is responsible for federated site management, and FATE Manager, a site client management terminal. It provides registration and management of federated sites, automated cluster deployment and upgrades, cluster monitoring, and permission control and other core functions.

- *FATE-Board* is a visualization tool for federated learning modelling, designed to deep explore models and understand models easily and effectively.
- *FATE-Flow* is actually FATE's workflow. It schedules and manages the lifecycle to build an end-to-end pipeline of federated-learning production services.
- *FATE-Serving* is a high-performance and scalable online federated-learning model-serving service that supports vertical federated learning cases.

Overall, *FATE* is a well designed and developed Federated Learning framework with great possibilities, especially for production. Nonetheless, it is highly resource-demanding therefore is not well suited for IoT devices and edge computing. Another major drawback of *FATE* is that currently, it does not support full deep learning model training. But the biggest drawback is that there isn't any English based detailed documentation for FATE's custom language (DSL) which makes it difficult to use it.

2.5.1.4 Flower

Flower [61] is an open-source federated learning framework which supports heterogeneous environments consisting of a few IoT devices and mobile phones while it is able to scale up to thousands of clients. *Flower's* architecture design assists engineers to integrate workflows from existing ML applications regardless of the ML/DL framework (PyTorch, TensorFlow, etc.) that they rely on with minimum performance overhead. In addition, facilitates research approaches due to its flexibility and interoperability. *Flower* is developed to meet some design goals which are listed as follows:

- *ML framework-agnostic* – This is a strong positive characteristic because ML frameworks are evolving rapidly and therefore possible alternations on running Federated Learning applications could lead to severe software crashes. *Flower* is compatible with almost any machine learning framework.
- *Client agnostic* – The fact is that typically, there is great structural heterogeneity among participants devices. Structural heterogeneity, can be a crucial drawback and to address that, *Flower* preserves interoperability with different operating systems, programming languages and hardware characteristics
- *Expandable* – Most Federated Learning frameworks lack expandability and this does not provide flexibility to upcoming research efforts. Flower aims to be expandable so that it can enable research and adopt recently proposed methods
- *Accessible* – Most Federated Learning frameworks tend to increase rapidly the engineering overhead in cases where ML applications need customization in order to run in a federated manner. *Flower* enables users to easily federate existing ML pipelines.
- *Scalable* – In real-world scenarios cross-device Federated Learning would encounter many clients. *Flower* is capable of scaling to a large number of clients.

In Figure 11, Flower's core framework architecture is presented. *Flower's* core framework is designed for being able to scale when workload increases. As the figure shows, *Flower* is divided into two, Server and Clients. The server-side includes three main components, the Federated Learning loop, the RPC server (the connection server) which is responsible also for sending and receiving Flower Protocol messages, and a *Strategy* which is user-customizable. Clients initially connect to the RPC server while the Federated Learning loops is the core federated learning process; basically, it acts as an orchestrator during the entire learning process.

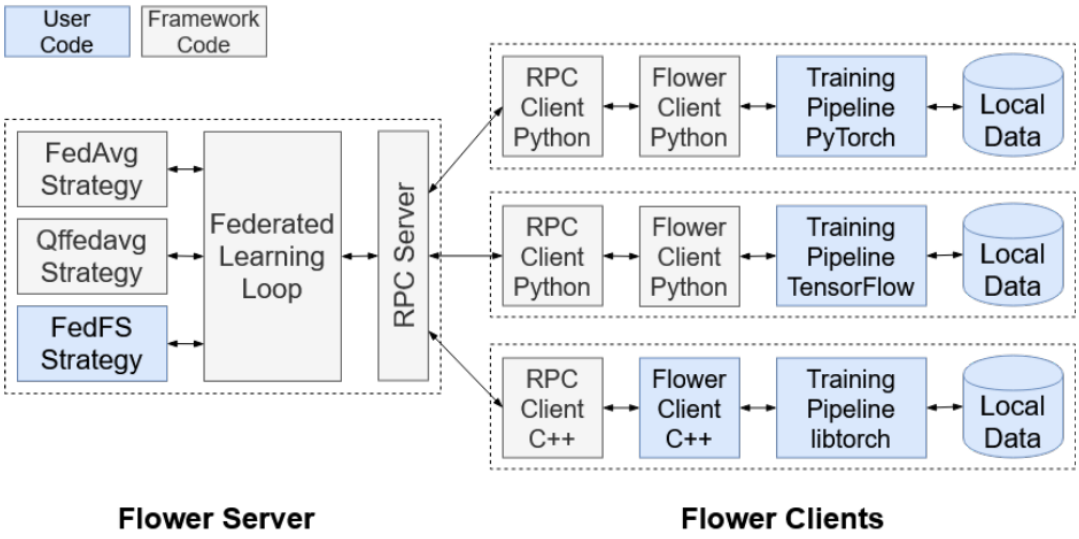


Figure 11: Flower core framework architecture, [61].

Strategy

Strategy is the main idea of *Flower*. As it was mentioned above, one of *Flower's* goals is to enable flexibility for researchers and application developers/engineers to orchestrate their pipelines in the way they want. This is achieved through a plug-in architecture which ensembles user-defined decisions to the abstract base class *Strategy*. The default *strategy* in *Flower* is the *FedAvg* but it is rather easy for users to implement their own.

Flower among others is one of the most well-suited for the IoT-NGIN platform because of its flexibility, customizability, interoperability, and easiness of use.

2.5.1.5 Sherpa.ai

Sherpa.ai is an open-source federated learning framework and was introduced in [62]. *Sherpa.ai* consists of seven different modules that encapsulate the functionalities for each key element in an FL setting, as shown in Figure 12, which are the following:

- *data_base* – this module is in charge of reading the data according to the chosen database
- *data_distribution* – in this module, the federated distribution of the data among participants takes place
- *private* – this module includes several interfaces such as the node interface which represents the client's key element and other ones for altering the federated data distribution
- *learning_approach* – this module defines software's structure to develop federated aggregation operators
- *model* – this module defines the machine learning model
- *differential_privacy* – this module is in charge of guaranteeing differential privacy to the participants

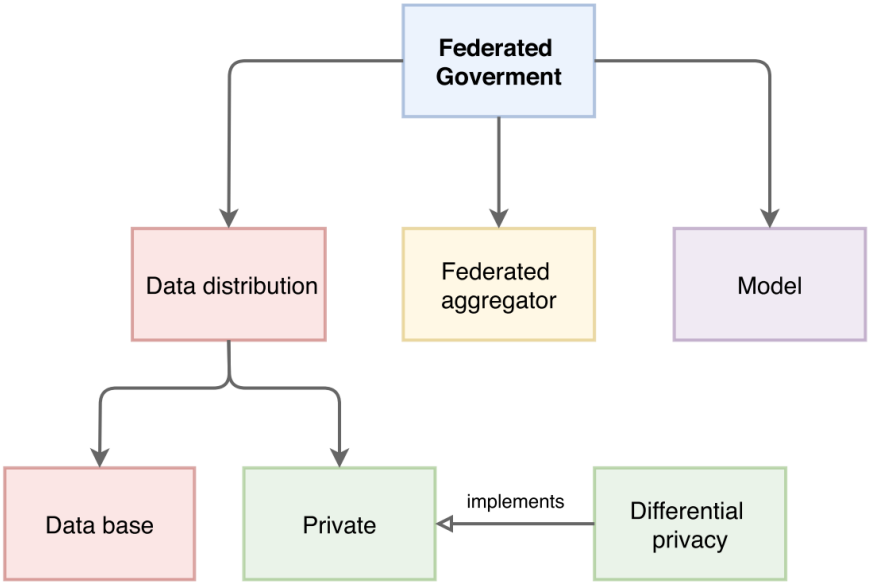


Figure 12: Overview of the Sherpa.ai module architecture, [62].

Sherpa.ai has strong privacy-preserving characteristics and it is easy to use but on the other hand, it lacks features that are essential in IoT-NGIN such as it can run only in simulation mode, and it has limited applicable scenarios.

2.5.1.6 FedML

FedML [63], is an open-source library and benchmark tool for Federated Learning projects which provides an end-to-end toolkit to support Federated Learning development and a legitimate performance comparison tool under diverse computing configurations. *FedML's* main design elements are as follows:

- *FedML* supports three different computing environments
  - on-device training for edge computing (mobile phones, IoT devices, etc.)
  - distributed computing with federated learning principles
  - single-machine simulation
- *FedML* presents a worker/client-oriented programming scheme to enable diverse network topologies, flexible information exchange and many training flows
- *FedML* provides standardized Federated Learning algorithms which help users to decrease the learning curve and also, to have a comparison baseline for newly developer algorithms
- *FedML* provides also standardized benchmarks with specific evaluation metrics, many synthetic and real non-IID datasets for comparison purposes

The architecture overview of *FedML* is presented in Figure 13 and consists of two key components, *FedML-API* and *FedML-core* which represents high-level API and low-level API, respectively.

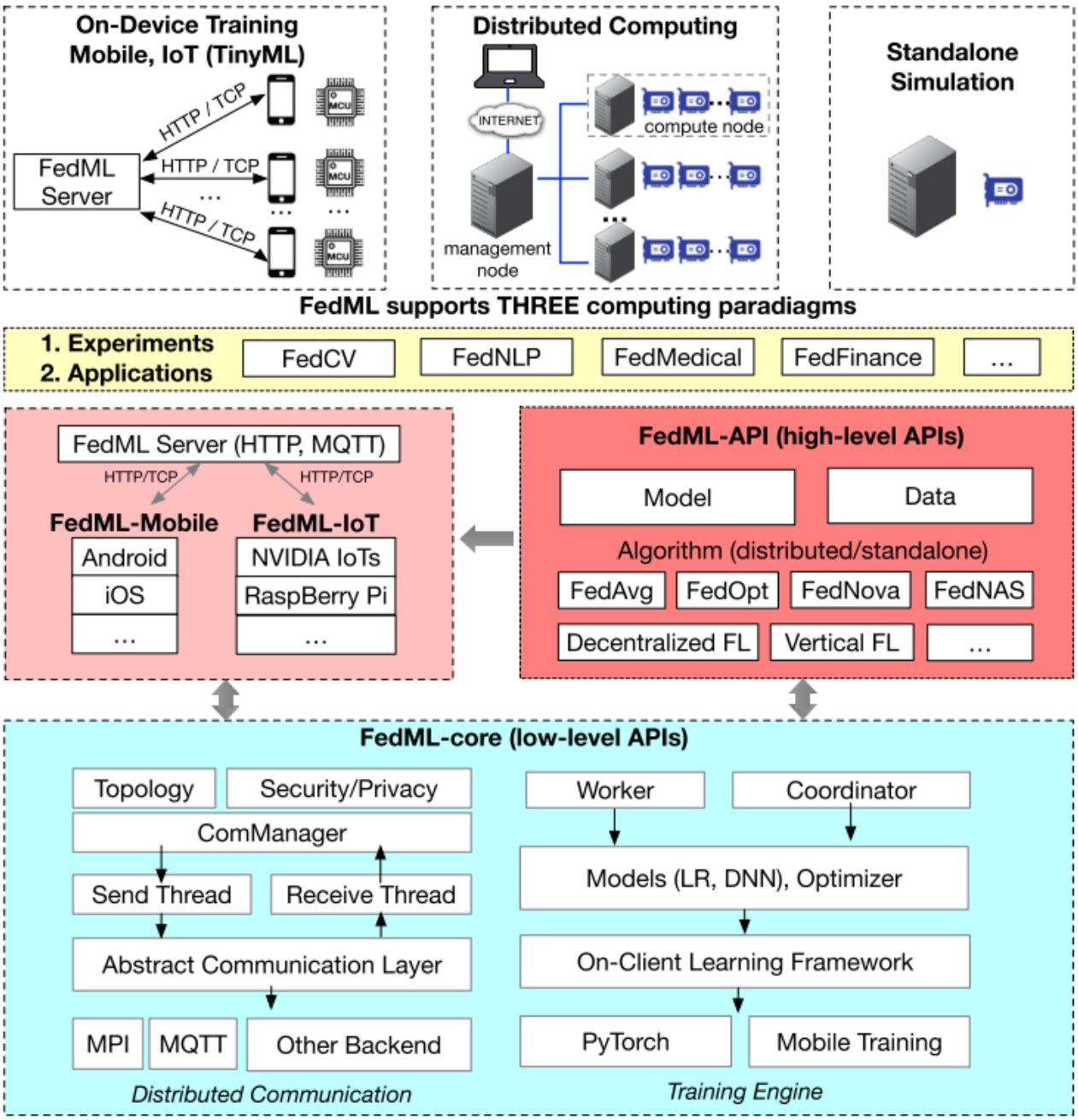


Figure 13: Architecture Overview of FedML and its components, [63].

FedML-core consists of two separate components, the distributed communication, and the training engine where distributed communication component is in charge of *workers/clients* low-level communication. *FedML-API* is built on top of *FedML-core*, and it is suitable for implementing new algorithms in a federated manner by adopting the client-oriented programming interface. In addition, *FedML-API* proposes a Machine learning system practice that separates the development of models, datasets and algorithms which facilitates reuse of implementations. *FedML* provides specific sub-tools (*FedML-Mobile, FedML-IoT*) for different real-world scenarios. The major drawback of using *FedML* is that currently doesn't provide privacy-preserving mechanisms.

2.5.1.7 PaddleFL

PaddleFL [64], is an open-source Federated Learning framework that can process horizontally and vertically portioned data. *PaddleFL* can be deployed in large scale clusters and supports several Federated Learning strategies targeting various application scenarios included but not limited to Computer Vision and Natural Language Processing.

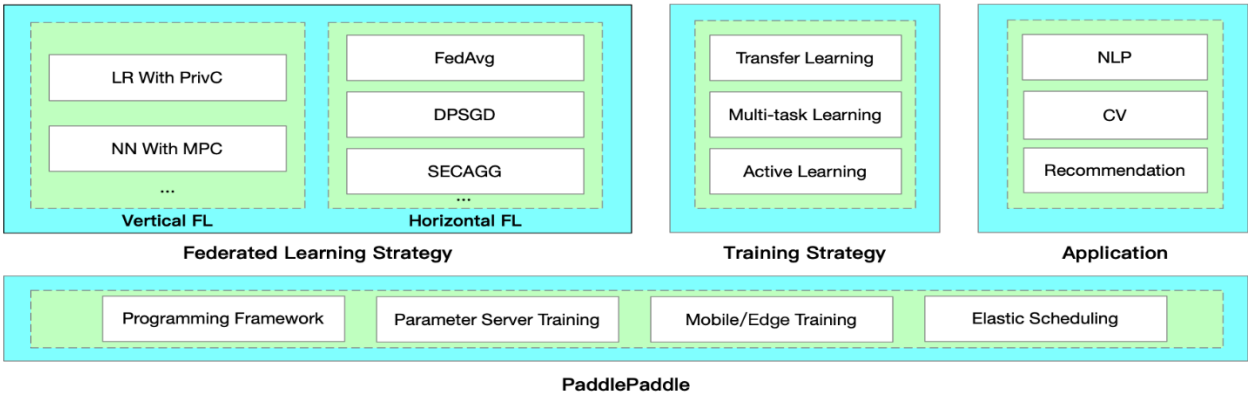


Figure 14: Architecture overview of PaddlePaddle deep learning platform, [64].

PaddleFL is based on *PaddlePaddle* (PArallel Distributed Deep LEarning) (Figure 14), which is a Deep learning platform. More specifically, it is an industrial platform with advanced technologies and includes features that can cover deep learning frameworks, basic model libraries, development kits and components. *PaddleFL* includes two main components:

- Data-Parallel – Using data parallelism, data owners can define their Federated Learning tasks based on common horizontal strategies like FedAvg, etc.
- Federated Learning with MPC – Enables secure training and prediction on vertical and horizontal datasets while supporting transfer Federated Learning

One of the most significant drawbacks is that *PaddleFL* uses a little-known Deep Learning platform (PaddlePaddle) with limited contribution from the community and has poor documentation.

2.5.1.8 LEAF

LEAF [65], is an open-source modular benchmarking Federated Learning framework tool. It includes several open-source datasets, an evaluation framework, and a set of Federated Learning algorithms implementations. *LEAF's* three main characteristics are that:

- Enables reproducible science
- Provides granular metrics
- Is modular

Like *PaddleFL*, *LEAF* has a small contribution team. It mainly focuses on benchmarking Federated Learning settings, but it doesn't provide any benchmarking tool for privacy-preserving.

2.5.1.9 OpenFL

OpenFL [66], is an open-source framework for training ML models in a Federated Learning manner and supports both TensorFlow and PyTorch while it can be easily extended to other ML and Deep Learning (DL) frameworks. *OpenFL* is designed to be project agnostic meaning that it can be used for many different application domains, such as healthcare and IoT. *OpenFL* consists of two core components, the Aggregator, and the Collaborator. A Collaborator node is a client that contains the dataset which is owned by a participant on which the learning model is applied. An Aggregator node is a compute node that receives locally tuned model updates from each collaborator and combines them into a new global model.

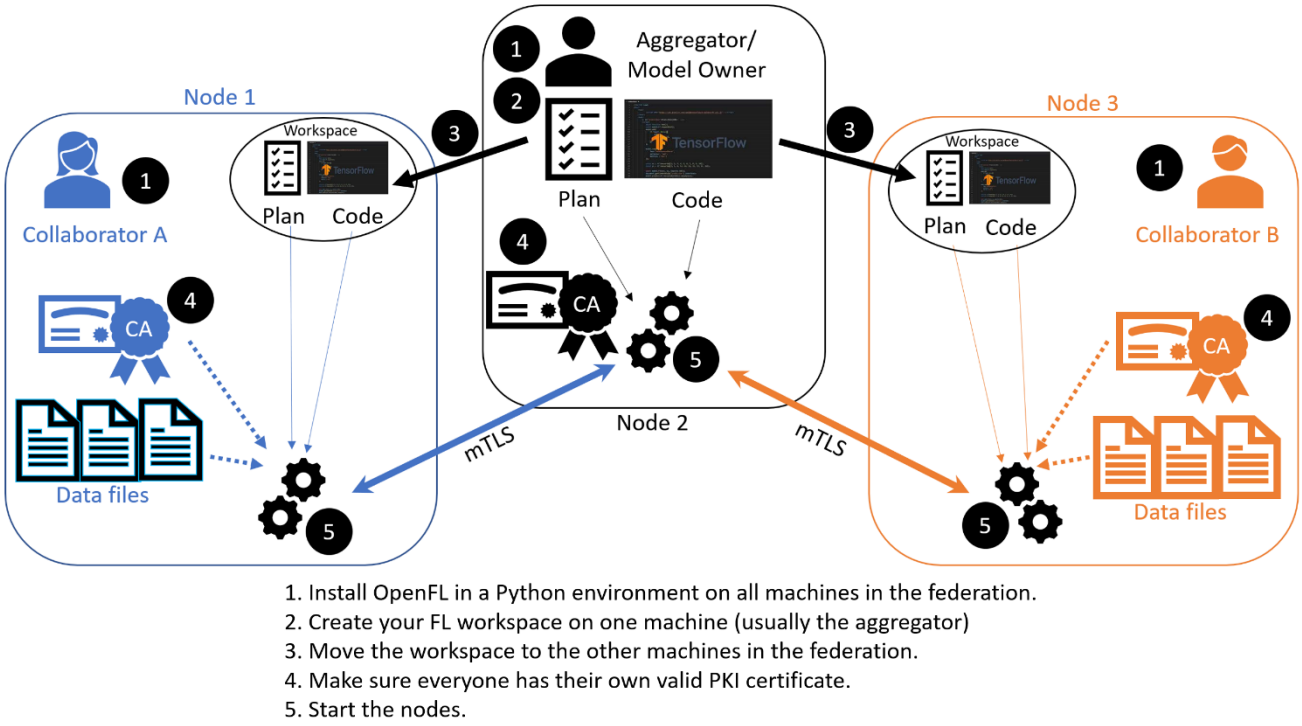


Figure 15: High-level workflow of OpenFL, [66].

Once *OpenFL* is installed on all computation nodes in the federated system and every client receives the PKI certificate, the workspace is distributed to every node. *OpenFL* creates a valid public key infrastructure certificate, the PKI which is created and signed by a trusted authority. Nonetheless, this certificate is not suitable for production tasks. The overall design of *OpenFL* relies on the *Federated Learning Plan* which is used to describe the configuration and workflow of the project. In particular, the *FL plan* is a YAML file that the tasks and the essential parameters required to coordinate and execute a federated task. It defines the collaborators, aggregator, connections, models, data, and any other parameters that describe how the training will evolve. In Figure 16, 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.

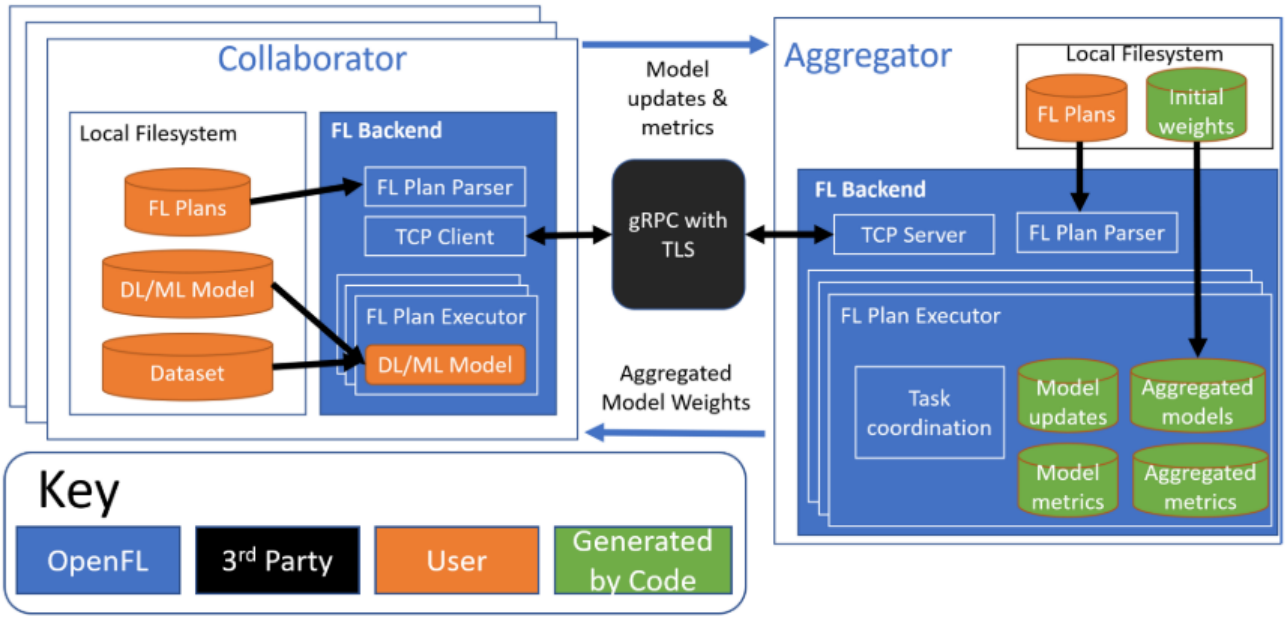


Figure 16: A high-level architectural overview of OpenFL with its components, [66].

2.6 Federated Learning aggregation algorithms

Federated Learning enables a set of participants to collaboratively train a model by sharing the model parameters rather than the data of each participant and has been largely researched in the literature with the aim of optimizing the distributed training process and costs. Federated Learning aggregation algorithms are the key mechanisms for updating the global model on the central server on each communication round or periodically. The federatedAveraging Algorithm (*FedAvg)* is a pioneering work for global model aggregation. In the next sub-sections, *FedAvg* along with some alternatives are presented.

2.6.1 The FederatedAveraging Algorithm

Federated Averaging (*FedAvg*) was introduced in [2] and is a global model aggregation algorithm which is executed in the central server once it receives the updated model weights from the clients. It uses simple gradient methods (*SGD*) which can be applied easily to a federated learning setup. Nevertheless, FedAvg performs more local computation and less communication compared to *SGD*. Moreover, *FedAvg* reduces the communication cost by choosing a fraction of clients on each training round and computing the *SGD* for these clients. *FedAvg* aims to minimize the objective of the global model w, which is the sum of the weighted average of the devices' loss:

$$\min_{w} F(w), \qquad where\ F(w) \coloneqq \sum_{k=1}^{n} p_k F_k(w) \tag{1}$$

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 $\sum_k p_k = 1$. The authors in [68] analyze *FedAvg* in-depth and demonstrate how partial client participation does not affect the learning process.

2.6.2 FedAvg alternatives

In this sub-section, some FedAvg alternatives are presented. Each Federated Learning aggregation algorithm tries to tackle a specific challenge as listed in section 2.1. Authors in [69] proposed *FedProx*, an algorithm which tries to address statistical heterogeneity across client devices. *FedProx* poses better results on non-identically distributed data while it provides more robust convergence than *FedAvg*. The main addition in *FedProx* is the introduction of a proximal term on each client device to improve the stability of the method, therefore limiting the influence of each local model updates on the global model. Proximal term provides a principled way for the server to account for heterogeneity associated with partial information. Similar to *FedAvg,* in *FedProx* all devices are weighted equally during global aggregation while hardware heterogeneity is disregarded. Reisizadeh et. al. in [70] presented *FedPAQ* (Federated Learning Periodic Averaging and Quantization) a federated learning aggregation mechanism which tries to reduce communication costs. *FedPAQ* inserts periodic averaging which means that models are trained and updated locally on the devices and periodically averaged at the server based on a parameter which corresponds to the number of local iterations. Similarly, to *FedAvg* only a fraction of clients participates in each training iteration. Another key feature is that *FedPAQ* quantizes each nodes' updates before uploading them on the central server. This quantization technique significantly helps reducing the communication overhead on the network. *Turbo-Aggregate* algorithm was proposed in [72] to reduce communication costs whilst enhancing security. *Turbo-Aggregate* employs a multi-group circular strategy for model aggregation. In particular, the clients are partitioned into several groups, and at each aggregation, model updates are shared in a circular manner among the groups. As the authors claim, this leads to the reduction of the aggregation overhead. *Turbo-Aggregate* introduces an additive secret sharing mechanism to preserve clients' data privacy thus enhancing privacy-preserving. Authors in [72] propose *HierFAVG*, an algorithm that allows multiple edge servers to perform partial model aggregation and aims at reducing communication costs. *HierFAVG* is based on a hierarchical client-edge-cloud architecture where its server updates its own clients. Global aggregation is done on edge-level aggregate models and takes place after a fixed number of model aggregations. *FedMA* [74] aims to address statistical heterogeneity in federated learning applications by proposing a layer-wise learning scheme, which includes the match and the merging of nodes with similar weights. The novelty in *FedMA* is that it considers the permutation invariance of the neurons in a neural network model before performing the aggregation. It further utilizes a Bayesian non-parametric method to facilitate adaptation to the global model size. *FedMA* poses better performance and communication efficiency than *FedAvg*. Nonetheless, the method works only for simple neural network architectures such as fully connected feedforward networks. In literature, there are also other approaches like [55], [74]–[76] for different aggregation mechanisms which one can refer to if a more extended review is needed.

Table 3: A summary of the Federated Learning aggregation mechanisms.

Federated Aggregation Method	Challenge to address	Summary
FedAvg	Statistical	FedAvg is a global model aggregation algorithm which is executed in the central server once it receives the updated model weights from the clients. It uses simple gradient methods (*SGD*). *FedAvg* reduces the communication cost by choosing a fraction of clients on each training round and computing the *SGD* for these clients.
FedProx	Statistical	*FedProx* poses better results on non-identically distributed data while it provides more robust convergence than *FedAvg*. Introduces a proximal term on each client device to improve the stability of the method. In *FedProx* all devices are weighted equally during global aggregation while hardware heterogeneity is disregarded.
FedPAQ	Communication	*FedPAQ* inserts periodic averaging from clients to the server based on a parameter which corresponds to the number of local iterations. Similarly, to *FedAvg* only a fraction of clients participates in each training iteration. *FedPAQ*, quantizes each nodes' updates before uploading them on the central server.
Turbo-Aggregate	Communication and Privacy	*Turbo-Aggregate* employs a multi-group circular strategy for model aggregation. Clients are partitioned into several groups, and at each aggregation, model updates are shared in a circular manner among the groups. It introduces an additive secret sharing mechanism to preserve clients' data privacy thus enhancing privacy-preserving.
HierFAVG	Communication	*HierFAVG* allows multiple edge servers to perform partial model aggregation and aims at reducing communication costs. *HierFAVG* is based on a hierarchical client-edge-cloud architecture where its server updates its own clients. Global aggregation

		is done on edge-level aggregate models and takes place after a fixed number of model aggregations.
FedMA	Statistical	*FedMA* inserts a layer-wise learning scheme, which includes the match and the merging of nodes with similar weights. It also considers the permutation invariance of the neurons in a neural network model before performing the aggregation. It further utilizes a Bayesian non-parametric method to facilitate adaptation to the global model size. *FedMA* poses better performance and communication efficiency than *FedAvg*.

3 Privacy-preserving Federated Learning

While FL is resilient and resolves, up to a point, data governance and ownership issues, it does not guarantee security and privacy by design. A lack of encryption can allow adversaries to abduct personally identifiable data directly from the processing nodes or interfere with the communication process, expose network vulnerabilities, and perform attacks. In addition, the decentralized nature of the data complicates data handling and curation. Moreover, in the case where algorithms running on the nodes are not encrypted, or the updates are not securely aggregated, the possibility of data leakage grows. Additionally, the algorithms can be tampered with, reconstructed, or get stolen (parameter inference), which can be strictly forbidden for most applications. Federated Learning can be vulnerable to various backdoor threats (bug injection, inference & model attacks) on different processing steps. Therefore, additional measures are essential to protect data from adversarial attack strategies such as data poisoning and model poisoning attacks. In Table 4, three major attacks against the dataset with their description and a basic example for each case, are listed while in Table 5, algorithmic-based attacks are presented.

Table 4: Various attacks against the data in a Federated Learning system.

Attacks against the dataset	Description	Example
Re-identification Attack	Recover an individual's identity by exploiting similarities to other datasets and exposing the data characteristics.	Exploiting similarities between data distributions and actual values from other datasets in which the same individual is contained.
Dataset reconstruction attack	Determine an individual's characteristics from the training process without accessing the data itself.	Using multiple statistical information (probabilities, distributions, etc.) to get data points that correspond to a single individual.
Tracing attack	Trace an individual to determine if it is present in the dataset or not without exposing their identity.	Exploiting repeated, slightly varying dataset queries to extract individual information and model the output.

Table 5. Major attacks against algorithms that run in a Federated Learning system

Attacks against the algorithm	Description	Example
Adversarial attack	Manipulation of the input to an algorithm with the goal of altering it, most often in a way that makes the manipulation of the input data impossible to detect by humans.	Compromising the computation result by introducing malicious training examples (model poisoning).
Model-inversion/reconstruction attack	Derivation of information about the dataset stored within the algorithm's weights by observing the algorithm's behaviour.	Using generative algorithms to recreate parts of the training data based on algorithm parameters.

In general, the goal of an adversary during data poisoning is to alter the data according to their preferences. This can be done by ingesting a mixture of clean and false data into the training flow. For example, in [77], the result of an image classification learning task can be vulnerable to a data poisoning attempt by a mislabeling or a false-labelling operation. Wang refers to different defence mechanisms from simple data management to more sophisticated and robust approaches. Data sanitization is a rather basic defence while pruning (removing neurons in a network) seems more reliable. Nonetheless, the pruning technique raises concerns regarding privacy-preserving in Federated Learning. In [78]-[80], some legitimate defences for these attacks are proposed, although backdoor attacks become stronger and more adjective. Model poisoning attack refers to partial or fully model replacement during training. Authors in [81], [82] describe possible attacks and argue about various defences (SMP, DP etc.). Generative Adversarial Networks (GANs) [83] can be one of the most vicious threats in Federated Learning. The authors in [84] exploit defences against GAN-based attacks and present the Anti-GAN framework to prevent adversaries from learning the real distribution of the training data. On the other hand, GANs in [85] are utilized as a defence mechanism against adversarial attacks in Federated Learning systems. As a conclusion, FL is vulnerable to various attacks and great attention must be given to the defence mechanisms and tools, otherwise, it will not be possible for an FL system to fulfil its privacy-preserving objectives.

3.1 State-of-the-art approaches in privacy-preserving Federated Learning

Although FL enables on-device machine learning, it does not guarantee security and privacy. The fact that the private data are not shared with the central server is for sure an

advantage, yet there are ways to extract private information from the data. After the shared model is trained on the user's device based on its own private data, the trained parameters (model weights) are sent to the central server and through an aggregation mechanism, the global model is composed. During the model transfer, it is possible for an adversary to extract information about the private data from those trained parameters. For example, in [86] the authors indicate that it is possible to extract sensitive text patterns, e.g., the credit card number, from a recurrent neural network that is trained on users' data. Therefore, additional mechanisms are required to protect data disclosure from attack strategies, which are subject to privacy-preserving methods in FL. The major approaches that can be employed in FL for data protection are differential privacy, homomorphic encryption, and secure multiparty computation.

Differential Privacy (DP) is a method that randomizes part of the mechanism's behaviour to provide privacy [87], [88]. The motivation behind adding randomness (either Laplacian or Gaussian) into a learning algorithm is to make it impossible to reveal data patterns or insights that correspond either to the model and the learned parameters or to the training data. Therefore, the DP provides privacy against a wide range of attacks (e.g., differencing attacks, linkage attacks) [89]. The method of introducing noise to the data can result in great privacy but may compromise accuracy. Therefore, there is a trade-off between applying differential privacy and achieving a high level of model accuracy. However, the authors in [89] present a method, which applies privacy-preserving without sacrificing accuracy.

Another privacy-preserving technique is the Secure Multiparty Computation (SMC), a well-defined cryptographic-based technique that allows a number of mutually suspicious parties to jointly compute a function before training a model while preserving the privacy of the input data [32], [90]. In the case of ML applications, the function can be the model's loss function at training, or it could be the model itself during inference. The challenge of applying SMC on a large-scale distributed system is the communication overhead, which increases significantly with the number of participating parties.

Homomorphic encryption [91] secures the learning process by applying computations (e.g., addition) on encrypted data. Specifically, an encryption scheme is characterized as homomorphic, when standard operations can be applied directly to the cypher data, in such a way that the decrypted result is equivalent to performing analogous operations to the original encrypted data [92], [93]. For machine learning methods, homomorphic encryption can be applied when training or inference are performed directly on encrypted data (cyphertexts). In scenarios, where large mathematical functions are implemented to cypher text space, a major bottleneck of homomorphic encryption emerges. The properties of homomorphic encryption schemes confront several limitations, related to encryption performance.

Alternative hybrid approaches that combine SMC with DP and account dishonest participants exist. In [94], authors confront the inference risk of SMC and the low accuracy that DP presents due to the noise injection by combining them. Furthermore, they propose a tunable trust parameter attribute by additively HE which considers many trust scenarios. HybridAlpha method [95] establishes a multi-input functional encryption (public-key cryptosystem) scheme to prevent inference attacks on SMC. HybridAlpha introduces a trusted third party to derive public keys to parties who intend to encrypt their data before training. Wang [96] presented HDP: a differential private framework for Vertical federated learning (cross-silo). HDP-VFL does not rely on HE or on third-party collaborators to assure data

privacy therefore, it is easy to implement and rather fast. Chain-PPFL [97] can achieve privacy-preserving without compromising the model accuracy using SMC and DP in a "trust-but-curious" way. The proposed communication mechanism constructs a serial chain frame which transfers masked information between participants. In addition, Chain-PPFL does not require encryption or obfuscation before transmitting information because it uses the P2P encrypted secure transmitted channel, thus requiring less resources. Authors in [98] present a fully decentralized federated learning process (BlockFlow) as a more resilient approach against adversarial and inference attacks. BlockFlow adopts blockchains as computational platforms and opposite to other methods does not require a central trusted part. Unlike other methods, there is no need for a centralized test dataset and different parties share DP models with each other.

3.1.1 Differential Privacy (DP)

Differential privacy provides strong guarantees for privacy by introducing randomness in the data, especially when the adversary has arbitrary external knowledge regarding the data. On the other hand, without considering randomness, adversaries could retrieve insights regarding the parameters that are required for the learning and convergence procedures on datasets or get the probabilities in which the learning algorithms will choose parameters within a set of possible learning parameters for a specific dataset. The use of differential private techniques eliminates such constraints. Generally, differential privacy may be divided into Local Differential Privacy (LDP) [99]–[101], and global differential privacy (GDP) [102].

LDP is a state-of-the-art approach which allows statistical computations while simultaneously protecting each individual user's privacy. No trust limitations to a central authority or a third party are necessary since noise is added to the individual inputs locally. For instance, consider the local nodes in the diagram posted in Figure 17. 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.

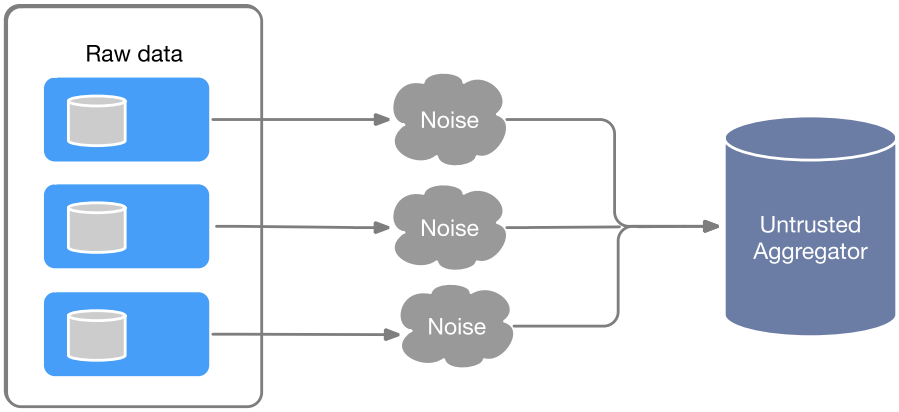


Figure 17: Local Differential Privacy.

In global differential privacy techniques, a central aggregator exists (i.e., a trusted curator) which has access to the raw data. In particular, each user sends their data to the aggregator node without adding noise. The aggregator then considers the input data and transforms it with a differentially private mechanism, by adding noise, either Laplacian or Gaussian. When an untrusted querier makes a specific query on the trusted aggregator node, an answer shall be provided, however, this answer is mathematically impossible to be reverse-engineered, and consequently it is impossible to know the precise answer about the private raw data. Generally, global private systems are more accurate, since all the analysis is implemented on "clean" (i.e. noise-free) data, and only a small amount of noise is added at the end of the process. However, the efficiency of global privacy models lies in the users' amount of trust in the trusted curator. Figure 18 overviews the general scheme of a global differential privacy example.

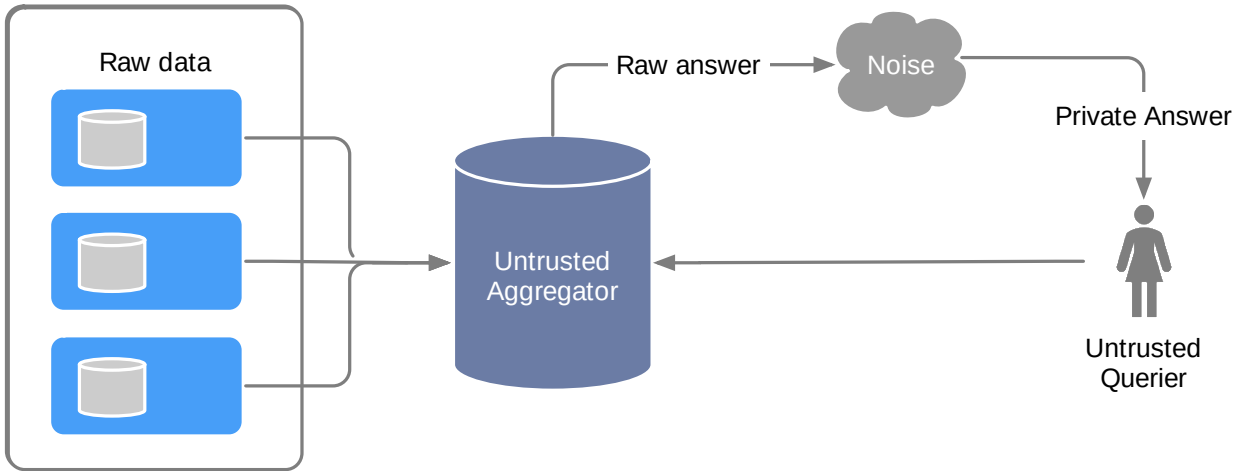


Figure 18: Global Differential Privacy.

Abadi et. al. [87] introduced a Differentially Private Stochastic Gradient Descent (DP-SGD) algorithm which aims to control the effect of the training data during the optimization operation (GD). For each step, the DP-SGD algorithm computes the gradient for a random set of data, calculates the clipped l_2 norm of each gradient computes the average, adds noise to preserve privacy and takes a step in the opposite direction of this SGD. Xu et. al. in [103] introduced an asynchronous decentralized parallel SGD with DP for FL which effectively reduces communication cost and does not rely on a central server thus, each computing node only communicates with its neighbours. Nevertheless, executing DP-SGD adds a significant runtime overhead. Subramani et. al. in [104] indicate some improvements such as vectorization, just-in-time compilation (accelerated linear algebra), and static graph optimization reducing the overhead up to fifty times. 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. Authors in [105], propose an enhanced robust DP mechanism for Federated Learning to address user dropouts as well as distributed noise generation and user-level privacy for differential privacy in Federated Learning.

3.1.2 Homomorphic Encryption (HE)

An encryption scheme is considered "homomorphic" if standard mathematical operations can be applied directly to the cypher text, in such a way that the decrypted result is equivalent to performing analogous operations to the original unencrypted data [92], [93]. For instance, on the diagram that is illustrated in Figure 19, we examine the sum operator on two values A and B. Starting from the private data, we encrypt each of these values with a key named P. Then, we consider the sum operator on the encrypted A and B values. This encrypted sum is able to be decrypted with the key P, and the result is equivalent to the original sum of the non-encrypted A, B values/ parameters [106]. However, in multiple scenarios, where larger mathematical computations/ functions are required to be implemented within the cypher text space, the remarkable properties of homomorphic encryption schemes confront several limitations, related to the encryption performance.

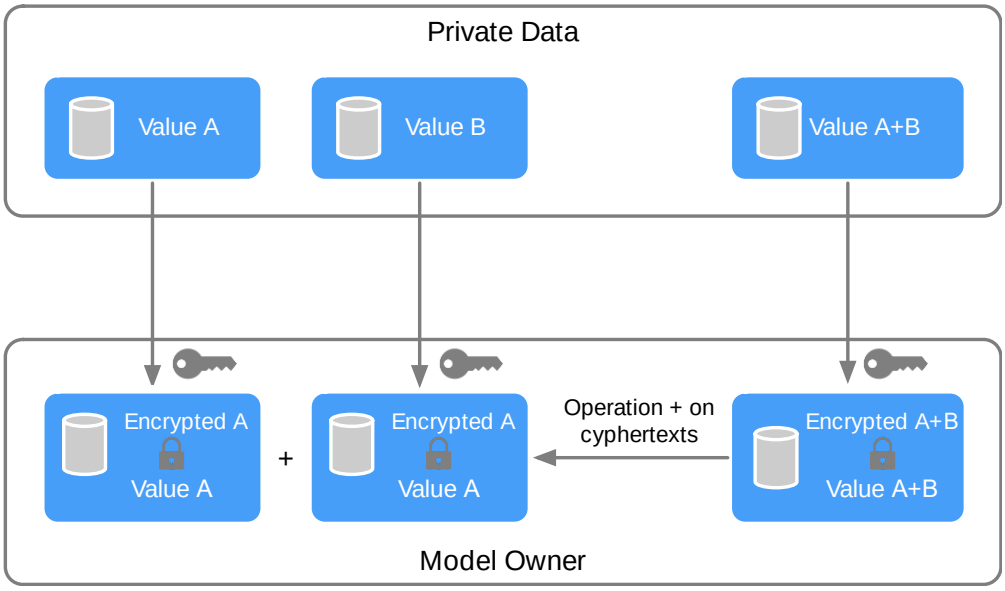


Figure 19: Homomorphic Encryption Example.

In this direction, multiple techniques have been proposed in the literature that overcomes these limitations by adopting statistical techniques to be compliant with homomorphic computation properties, and by quantifying reasonable approximations in those situations where a traditional approach cannot be implemented homomorphically [107].

[108] indicates that more than 80% of the training iteration time is spent on encryption/decryption when standard HE is used. CryptoNets [109] replaces the activation function of the neural network with the lowest-degree non-linear polynomial function to make homomorphic encryption feasible. The authors in [110] adopt a scale-invariant method through bootstrapping (ciphertext refreshing technique) and propose the use of discretized neural networks. Scale invariance within the network means that what happens in neuron's level does not depend on the size of the neural network. Therefore, facilitates the

use of Fully-HE for deep neural networks. On contrary, due to the discretization of the network, Bourse's approach underperforms in accuracy, compared to CryptoNets. The authors in [111] present "Faster-CryptoNets", an optimized method of CryptoNets, for efficient encrypted inference towards a HE-enabled neural network model, achieving significant speed improvement whilst proposing a quantized polynomial approximation to the activation function. Jukevar in [112] presents Gazelle, a low latency framework for secure neural network inference which uses HE with garbled circuits and linear algebra kernels. Gazelle is constructed by three components, the Gazelle Homomorphic Layer (GHL), the Gazelle Linear Algebra Kernels (GLAK) and the Gazelle Network Inference (GNI). Gazelle utilizes innovative calculation techniques on HE, leading to accelerated HE operations.

3.1.3 Secure Multiparty Computation (SMC)

Secure Multiparty Computation (SMC) is a well-defined cryptographic technique that enables distributed parties to jointly compute an arbitrary function without revealing their own private input and output pairs [32], [90]. SMC models provide security verification in well-defined simulation frameworks that guarantee complete zero-proof knowledge[113]. Specifically, using universally verifiable Zero-Knowledge Proofs (ZKPs) ensure computation integrity, and prove that the encrypted data are within the appropriate/given ranges. Nevertheless, in multiple scenarios, this desired property requires complex computations which are not very efficient. Additionally, several use-cases consider as acceptable the disclosure of partial knowledge under the constraint that pre-defined security guarantees are provided. For instance, the authors in [114] present an SMC framework for training machine learning models for linear regression, logistic regression and neural network training using the stochastic gradient descent method, with two servers and semi-honest assumptions. Additionally, in [115], the authors investigate the usage of Multi-Party Computation (MPC) protocols for machine learning model training, by encrypting sensitive attributes. Using this strategy, they demonstrate how an outcome-based fair model can be trained and verified without being subject to revealing its sensitive attributes. Consequently, the above-mentioned state-of-the-art approaches guarantee privacy even in the scenario of having semi-honest or malicious assumptions. Yet, in practical systems, algorithms tend to be simpler and more pragmatic while preserving the necessary accuracy.

3.1.4 Private Aggregation of Teacher Ensembles (PATE)

A well-known approach is the Private Aggregation of Teacher Ensembles (PATE) strategy [116], which presents a teachers-student scheme. Teachers are basically, the unpublished models trained on disjoint partitioned datasets with sensitive data. In Figure 20, PATE's general architecture is presented.

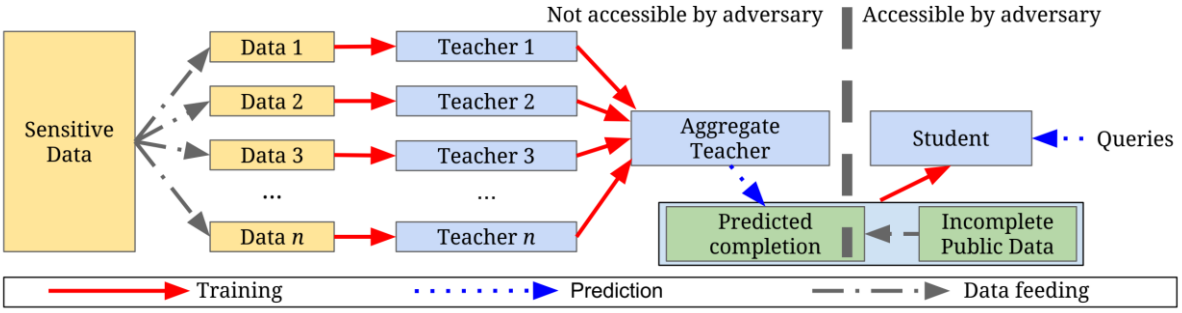


Figure 20: Approach diagram: an ensemble of teachers is trained on disjoint subsets of the sensitive data and a student model is trained on public data labelled using the ensemble [116].

A student cannot directly access the teacher's data or parameters and learn to predict, based on public non-sensitive data, an output which relies on noisy voting among all the teachers. A student model is trained in a semi-supervised way with GANs [82]. GAN framework consists of two machine learning models, a generator which produces samples from the data distribution and a discriminator which is trained to distinguish fake samples (produced by generator) with samples of the real data distribution. It is important to limit students' access to its teachers and essential to maintaining privacy, thus DP (Differential Privacy) is exploited. In general, DP is defined using pairs of adjacent inputs and PATE introduces a function to compute the privacy loss and the privacy loss random variable based on DP. To preserve privacy, the privacy cost of each outcome queried by the student should be bound. The total cost of training the student derives by exploiting the strong composition theorem [117]. To keep track more efficiently of the privacy cost, PATE exploits moments accountant, a technique which bounds the privacy cost when the topmost vote has a large quorum. Oppositely to typical machine learning techniques, PATE partitions the data in disjoint sets and trains a specific model for each set. Then, an aggregated teacher queries each teacher for a prediction and aggregates the output to a single prediction. This aggregation mechanism guarantees privacy. Nevertheless, if two classes have close vote counts, the disagreement may reveal private information. Therefore, PATE introduces random Laplacian noise (LNMax) to the vote counts and outputs the noisiest votes from the teacher's aggregator. To this matter, Papernot in [117] indicates an improvement in the teacher aggregator mechanism. Instead of using LNMax Papernot applies GNMax, a Gaussian based noise distribution which is more concentrated than the Laplace distribution. This modification improves aggregation's utility when the number of teachers is large whilst reducing the amount of noise needed to achieve the desired privacy cost per student query. GNMax relies on the Rényi Differential Privacy or RDP [119]. Papernot accounts for the close relation between RDP and moments accountant and proposes RDP as a more natural analysis framework when Gaussian noise is applied.

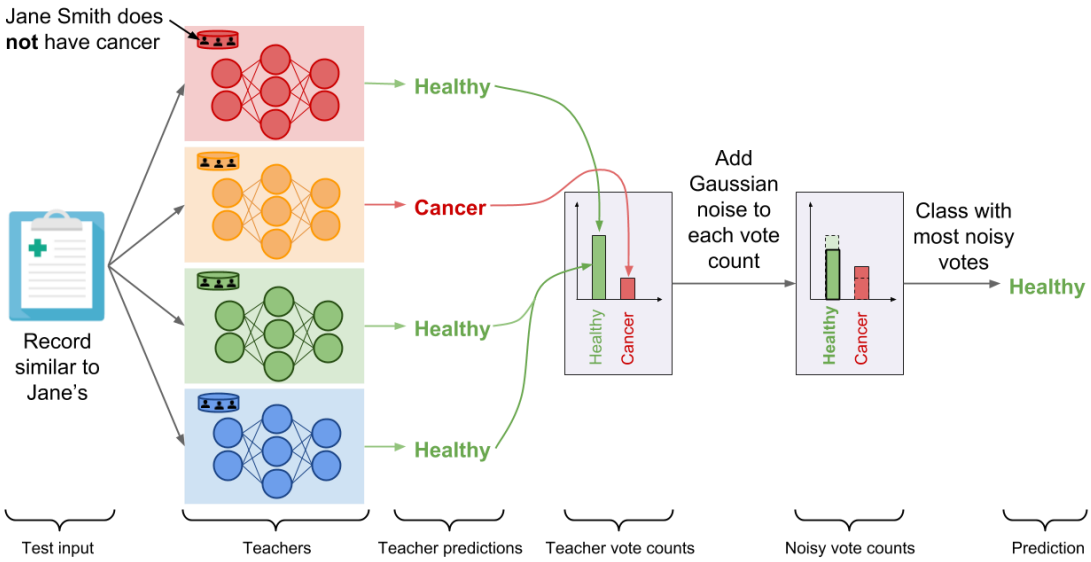


Figure 21: PATE, a detailed overview of the aggregation mechanism which shows that differential privacy does not change the label of an output, [120].

In Figure 21, 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.

3.2 Comparison of Federated Learning frameworks considering privacy preservation

Considering the extensive analysis presented in sections 2 and 3, for the FL methods/tools and the privacy-preserving approaches, respectively, comparative analysis for Federated Learning frameworks is conducted and presented in this subsection. The comparison refers to the FL frameworks analyzed in section 2.5 and for which the main benefits and drawbacks are briefly presented in Table 6.

Table 6: Main pros and cons of the Federated Learning frameworks.

FL Framework	Main Pros	Main Cons
FATE	1. Production Ready 2. High-Level interface 3. Provides many FL algorithms 4. Containerized - Kubernetes support	1. It does not establish any differential privacy algorithms 2. Its high-level interface relies too much on a poorly documented domain-specific language 3. Does not have a core API so developers must modify the

		source code of FATE to implement custom FL algorithms 4. Doesn't use GPUs for training 5. Not well suited for the purposes of the IoT-NGIN
Flower	1. Provides a template API which allows users to easily transform ML pipelines to FL 2. Very easy to develop and ML framework-agnostic 3. Supports a great number of clients 4. It is really customizable 5. Easy integration with privacy-preserving frameworks (PATE)	1. It does not have any differential privacy algorithms. 2. It is relatively new and the support community is not that big 3. Doesn't provide secure aggregation
PySyft & PyGrid	1. Rather easy to use 2. It has the largest community of contributors among the FL frameworks.	1. PySyft is only for 1 server and 1 client (Duet) and can run only in simulation mode 2. You need PyGrid in order to develop real FL scenarios
TensorFlow Federated	1. It integrates seamlessly with existing TensorFlow ML models 2. It is easy to use due to its familiarity	1. Until now It can be used only in simulation mode because it doesn't support the federated operation mode 2. The data used for training cannot be loaded from the remote worker itself, but must be partitioned and transferred through the central server
Sherpa.ai	1. Relatively easy to use because of the Jupiter notebooks etc. 2. Implements FL algorithms and it's easy to customize them	1. Poor Documentation. 2. Small community with only seven contributors 3. The Project's repository is not active (4+ months after the latest update) 4. Can run only in simulation mode 5. Limited applicable scenarios

FedML	1. On-device training for edge devices including smartphones and Internet of Things (IoT) 2. Distributed computing 3. Growing community 4. Multi-GPU training support	1. No privacy-preserving techniques are applied. Only a secure aggregation technique is implemented. 2. The multiple available modules for different situations might lead to drawbacks and create overheads
PaddleFL	1. Provides a high-level interface for some basic and well-known FL aggregators and implements a differentially private algorithm. 2. Provides enough privacy-preserving methods such as DP, MPC and secure aggregation	1. It is fairly difficult to use it because it uses a little-known DL platform. 2. Has poor documentation and has a small community—only 12 contributors. 3. It is not compatible with other frameworks and that is a major drawback
Leaf	1. Provides some basic Federated Learning mechanisms such as the Federated Averaging Aggregator 2. It is modular and adaptive 3. Enables reproducible science	1. It does not provide any benchmark for preserving privacy in an FL setting 2. Does not offer as much official documentation or tutorials 3. Limited Federated Learning capabilities, it is mainly for production purposes

The comparison among the FL frameworks listed in Table 6 is based on the following criteria:

- *Criterion 1*: This criterion is based on Basic Federated Learning features. The operating system support, the Federated Learning categorization e.g., if it supports cross-silo or cross-device setups, which machine learning and deep learning libraries (TensorFlow, PyTorch) do the framework supports and if there is a Federated attack simulator.
- *Criterion 2*: This includes three computing paradigms; the standalone simulation which gives the possibility for a user to apply FL scenarios in simulation; the distributed computing capability that shows if an FL framework is capable of performing in a distributed environment where participants are different devices; the capability of on-device training for IoT and other mobile devices which normally have limited hardware resources.
- *Criterion 3*: If FL frameworks include standardized FL algorithms and configurations like Federated Average, FedNAS [119], decentralized FI, vertical FI, and split learning [120].
- *Criterion 4:* An essential characteristic for an FL framework is the existence of privacy-preserving mechanisms and also, what types of privacy-preserving methods are supported by the frameworks. In cases where privacy-preserving techniques are not presented, the FL framework must give the capability to integrate such mechanisms.

- *Criterion 5:* In order for an FL framework to be flexible and adaptive, documentation, tutorials and Community support are significant.
- *Criterion 6:* Secure aggregation [121] algorithm implementation to further enhance privacy.
- *Criterion 7:* Nowadays, training on GPUs especially for Deep Learning tasks is essential. Especially for limited hardware resources on devices GPUs have shown remarkable computation capabilities compared 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:* More general properties and characteristics of the FL frameworks. To be more specific, an FL framework must be easy to use, adaptive, preserve interoperability, flexibility, and privacy.

The characteristics of each FL framework against the nine identified criteria are tabulated in Table 7 and Table 8.

Table 7: Federated Learning framework comparison (1/2).

FL Framework	FATE	Flower	PySyft + PyGrid	TensorFlow Federated
standalone simulation	Yes	Yes	Yes	Yes
distributed computing	Yes	Yes	Yes	Yes
on-device training (Mobile, IoT)	No	Yes - Depends on the Network	Yes	No
FedAvg	Yes	Yes	Yes	Yes
Decentralized FL	No	Yes	No	No
FedNAS	No	Yes	No	No
Vertical Federated Learning	Yes	Yes	No	No
Split Learning	No	Yes	Yes	No
Privacy-preserving Methods	No	Differential Privacy (PATE) - Implemented in IoT-NGIN	Multi-Party Computation - Homomorphic Encryption	Differential Privacy
DP Noise type	No	Yes	No	No
Adaptive Differential Privacy	No	No	No	No
Subsampling methods to increase privacy	No	No	No	No
Documentation and Community support	Partial - Mostly in Chinese	Yes, and it's growing rapidly	Yes	Yes
Secure Aggregation	Yes	Future Implementation		No

Table 8: Federated Learning framework comparison (2/2).

FL Framework	Sherpa.ai	FedML	PaddleFL	OpenFL
standalone simulation	Yes	Yes	Yes	Yes
distributed computing	No	Yes	Yes	Yes
on-device training (Mobile, IoT)	No	Yes	No	No
FedAvg	Yes	Yes	Yes	Yes
Decentralized FL	No	Yes	Yes	Yes
FedNAS	No	Yes	No	No
Vertical Federated Learning	No	Yes	Yes	Yes
Split Learning	No	Yes	Yes	Yes
Privacy-preserving Methods	Differential Privacy	No	Multi-Party Computation & Differential Privacy	Multi-Party Computation & Differential Privacy
DP Noise type	Yes	No	Yes	Yes
Adaptive Differential Privacy	Yes	No	Yes	Yes
Subsampling methods to increase privacy	Yes	No	Yes	Yes
Documentation and Community support	Yes	Stable	Partial	Partial but growing
Secure Aggregation	No	Future Implementation	Yes	Yes

Based on the tables above, privacy-preserving methods are available for Flower, PySyft & PyGrid, TensorFlow Federated, Sherpa.ai, PaddleFL and the relatively new OpenFL; however, the exact Privacy-preserving methods supported differ across the FL frameworks. On the other hand, Sherpa.ai, PaddleFL and OpenFL do not support on-device training, which would be essential under the (far-)edge computing paradigm adopted also in some IoT-NGIN LL use cases. Considering these, as well as the support of rest features, *Flower*, *PySyft & PyGrid* and *TensorFlow Federated* enhanced with privacy-preserving methods are identified as the most suitable candidates for being further analyzed and supported within the IoT-NGIN project.

4 IoT-NGIN privacy-preserving FL framework

Based on the analysis and comparison among eight FL frameworks of the previous section, there is no single solution that fits everything, in the sense that different frameworks may be most appropriate for different use cases. As a result, the design of the IoT-NGIN privacy-preserving federated learning framework is presented in this section towards providing privacy-preserving FL as a service. Indeed, it constitutes a generic approach, which may integrate and support multiple diverse FL frameworks. IoT-NGIN has identified three of them, namely Flower, PySyft together with PyGrid and TensorFlow Federated, as best candidates for covering diverse use cases, as the ones considered for the project validation. The initial version of the IoT-NGIN privacy-preserving FL framework incorporates PATE implementation in Keras and the Flower framework. Moreover, the high-level architecture design of integrating PATE with the Flower FL framework is presented and will be implemented as part of the future IoT-NGIN activities.

4.1 Design of IoT-NGIN privacy-preserving FL framework

Different privacy-preserving FL frameworks may be appropriate for different use cases, while the Federated Learning setup may be affected by different network structures involving IoT, edge or cloud nodes that may act as FL clients or as the FL server (or FL Aggregator). The high application and device diversity under the cloud, edge, fog continuum, combined with the varying characteristics of different FL frameworks, may highly complicate the ML developer's job in choosing a proper one.

IoT-NGIN aims to tackle this, by facilitating ML developers to use an appropriate FL technique to train their models, without having to delve into the technique's specifics. IoT-NGIN will basically support different FL frameworks, for which it will provide guidance for the user (ML developer) on what to choose, assisting, thus, developers' decision as per the suitable FL framework for training their models. The recommended FL framework will be able to be deployed on the designated (by the ML developer) nodes via IoT-NGIN's continuous deployment tools. In this way, IoT-NGIN will support privacy-preserving Federated Learning as a Service (PPFLaaS).

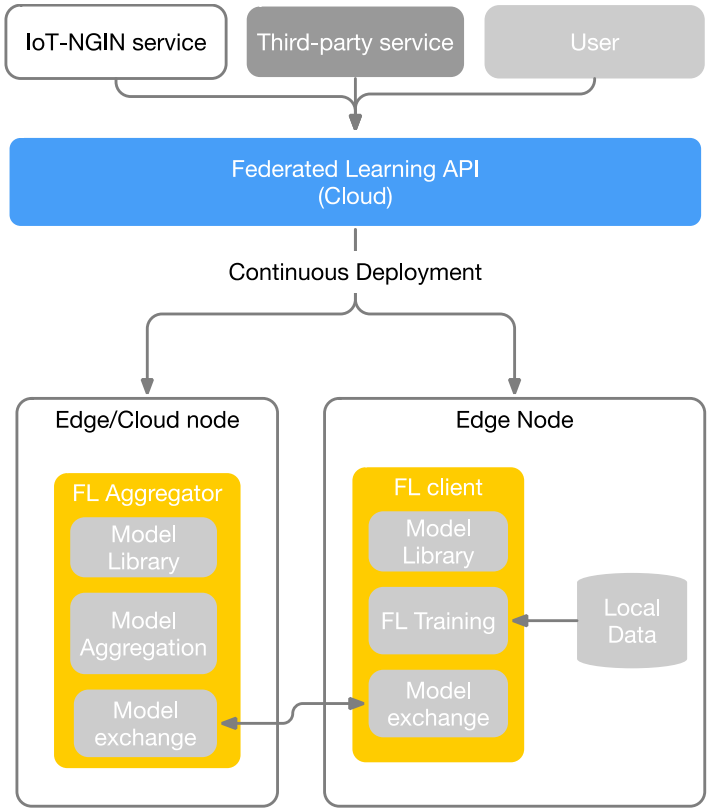


Figure 22: High-level architecture of IoT-NGIN as a Service.

The PPFLaaS concept of IoT-NGIN can be materialized via the high-level architecture presented in Figure 22. As depicted in the figure, PPFLaaS may be accessed through the *Federated Learning API*. This API will be responsible for suggesting the most appropriate privacy-preserving FL framework. The selection of the method will be conducted on the basis of the requester input, with respect to a set of predefined criteria, similar to the ones presented in the comparative analysis of section 3.2. The selected FL framework will be deployed on the nodes, also designated by the requester, via IoT-NGIN Continuous Deployment tools. Following the edge computing paradigm, FL clients would be deployed on edge nodes, while the FL Aggregator would be deployed on an edge or a cloud server.

4.1.1 FL in IoT-NGIN MLaaS

IoT-NGIN designs and develops a Machine Learning as a Service (MLaaS) platform that will facilitate access to the services and functionalities required to build powerful Machine Learning models, covering their complete life cycle (MLOps) and integrating the most powerful and recent advances in the state-of-the-art like deep learning, online learning, reinforcement learning, polyglot model-sharing and of, course, federated learning.

Thus, the IoT NGIN MLaaS platform will reduce the complexity that must be addressed currently to provision and support the hardware and software components used in the development of modern AI-based services, boosting time-to-market and even democratizing the access to AI technologies for any company independently of its size.

One of the main principles guiding the design of the IoT NGIN MLaaS platform is the need to seamlessly integrate the new generation of systems resulting from the increasing capabilities

and relevance of edge devices following the computing continuum paradigm. In this vision, IoT sensors and devices are not anymore just mechanisms to obtain high-quality data or to execute certain actions. They have become indeed intelligent nodes with the capacity to process at the edge collected information, reducing the amount of information that must be uploaded in centralised cloud platforms and opening new use-cases in multiple domains. In this sense, Federated Learning is clearly one of the points highlighted in the convergence between the new generation of IoT systems based on powerful edge computing devices and Artificial Intelligence technologies. Thus, its integration as part of IoT NGIN MLaaS platform is a fundamental requirement.

A complete and detailed specification of IoT-NGIN MLaaS platform is included in deliverable D3.1 "Enhancing deep learning / reinforcement learning" [130]. In this document, a high-level architecture has been proposed resulting in the diagram included in Figure 23.

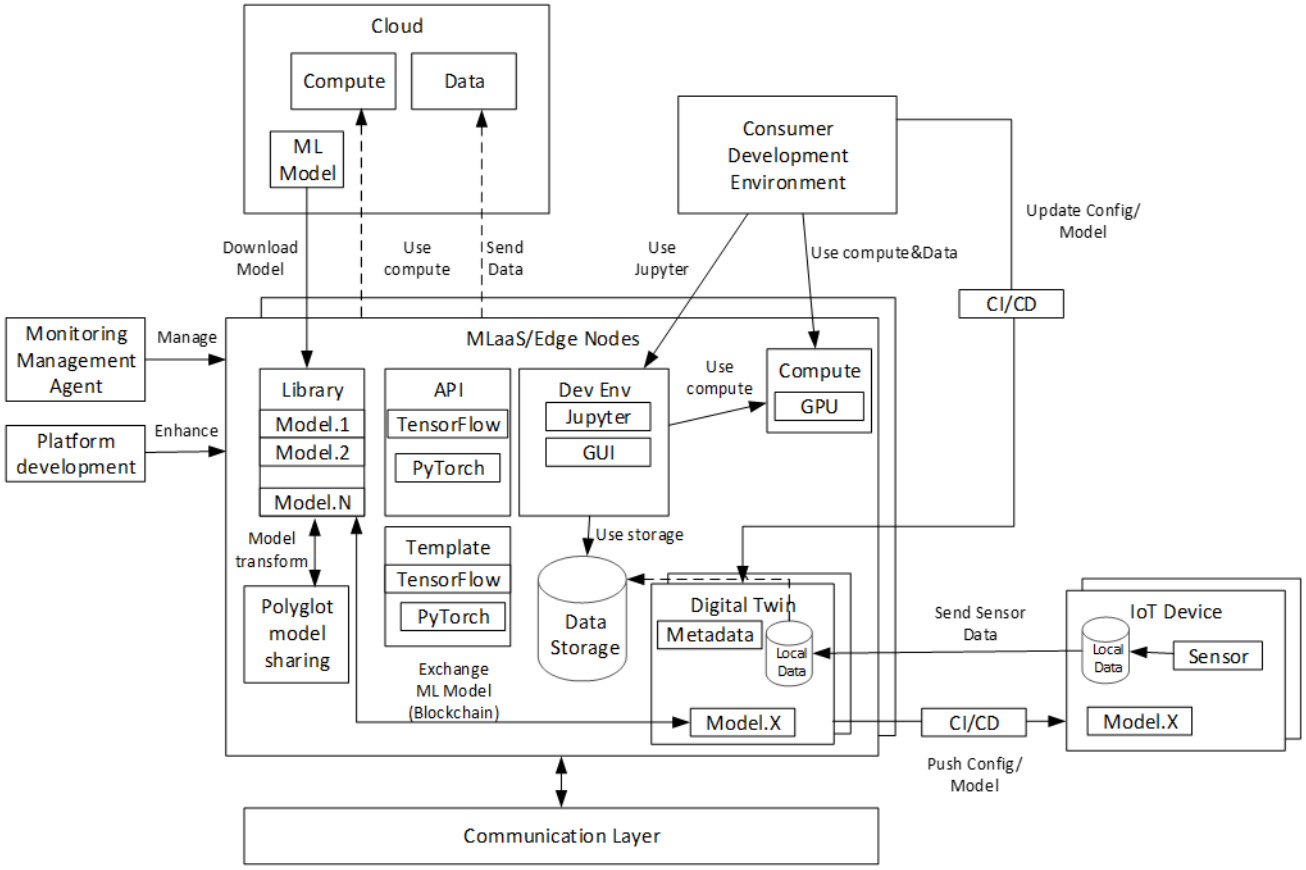


Figure 23: MLaaS high-level architecture.

As can be seen, IoT NGIN MLaaS platform design is focused on the provision of a complete ML environment that will be tailored to be deployed on edge computing nodes. A central cloud platform is considered as a complementary element to allow offloading the processing of certain tasks, to download and publish models and data that can be shared between several instance of the MLaaS platform. This approach perfectly fits with the PPFLaaS concept described in the previous subsection 4.1; the aggregation role will be played by the edge node for configurations where IoT devices or Digital Twins are used to deploy FL clients, but it will be possible to have a higher level of aggregation between several MLaaS instances thanks to the presence of the cloud platform.

With respect to the integration of the FL techniques widely described within the present document into the overall MLaaS platform, it is especially relevant to highlight the technical decisions that have been made from the release of D3.1 in terms of selection of technologies since they have a deep impact on this aspect. After concluding the analysis of relevant existing open-source solutions that could be used as baseline for the development of the MLaaS platform, the consortium made the decision to rely on Kubeflow[1], an open-source framework to build Machine Learning pipelines including training and serving phases on any infrastructure managed with Kubernetes. Kubeflow has a huge an active community and it is released with an Apache 2.0 license, so it is a perfect choice in order to maximise the long-term sustainability of IoT NGIN MLaaS and to maximise its impact and outreach. Figure 27 presents the Kubeflow platform architecture.

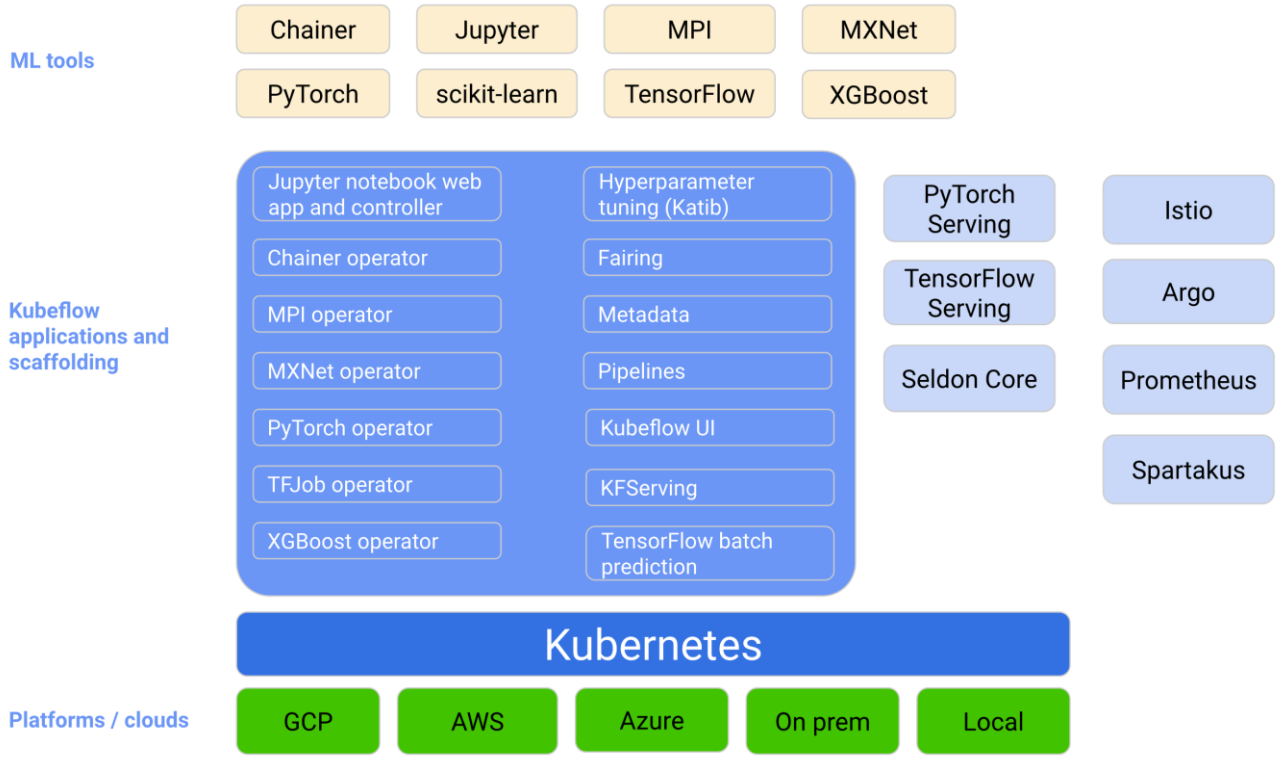


Figure 24: Kubeflow platform architecture. Source: https://www.kubeflow.org/docs/started/architecture/

As can be seen, Kubeflow relies on Kubernetes to enable the seamless and scalable deployment of ML workflows on multiple platforms covering mainstream cloud solutions for Infrastructure as a Service (IaaS), Infrastructure as a Service (PaaS) but also on-premises configurations.

One of the most interesting characteristics of Kubeflow from the perspective of a data scientist of ML engineer is the possibility to create complex workflows embracing MLOps practices that span from experimentation to production phases. Especially in the latest stage, Kubeflow includes functionalities to service the model (e.g., KFServing, Seldon Core) and to monitor its performance triggering a retraining process if needed. The solution supports well-known and widely used frameworks and libraries like TensorFlow, PyTorch, scikit-

[1] https://www.kubeflow.org/

learn or NVIDIA TensorRT. It also integrates off-the-shelf Jupyter notebooks, which are a de-facto standard for the initial stages of the development of ML models. A deeper analysis of Kubeflow will be included in the deliverable D3.3 "Enhanced IoT federated deep learning/ reinforcement ML", due at the end of Q3 2022.

Regarding the integration of Federated Learning techniques into Kubeflow, there are not many available implementations that can be considered. A specific operator for FATE (see section 2.5.1.3) has been proposed by the community developed by VMware as open-source [2]. The architecture for this implementation is showed in Figure 25.

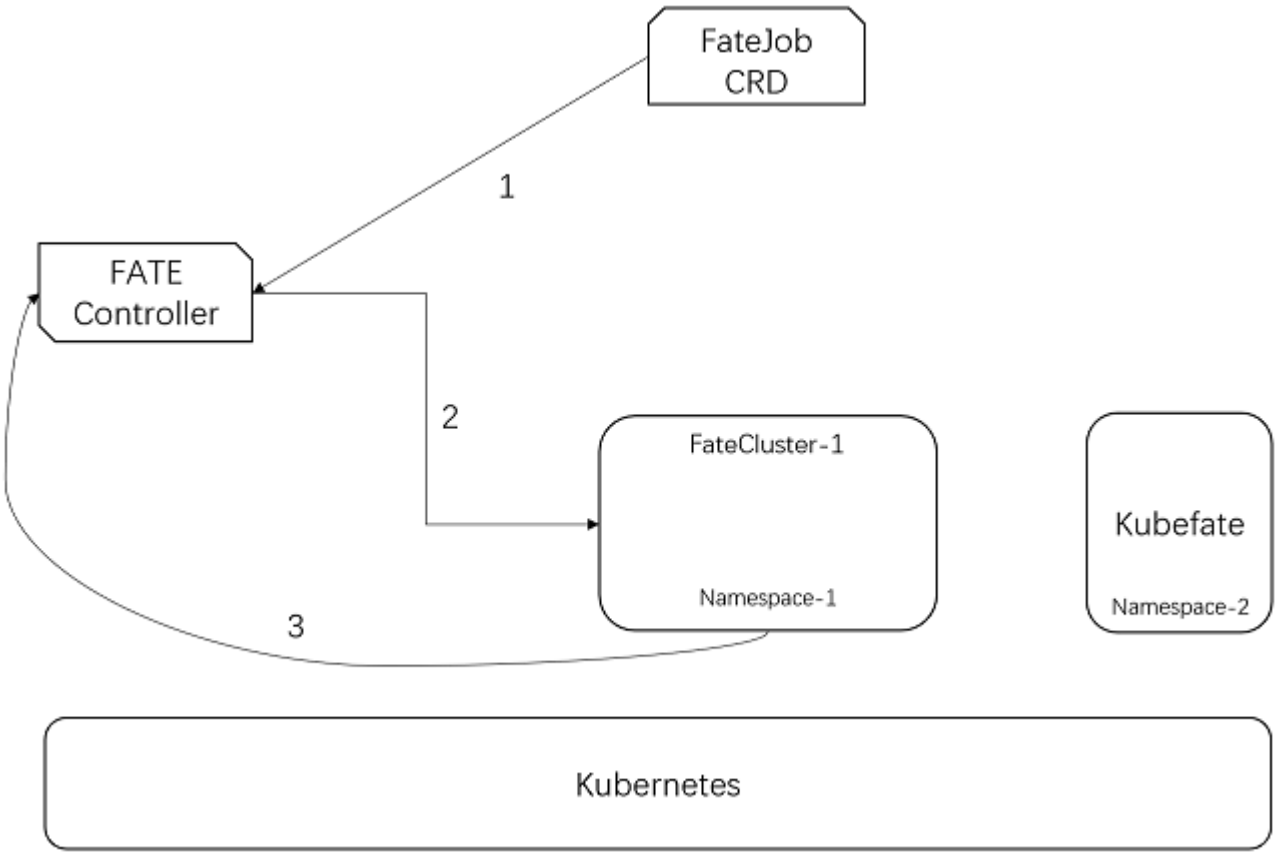


Figure 25: Architecture of FATE Kubeflow operator. Source: https://github.com/kubeflow/community/blob/master/proposals/fate-operator-proposal.md

KubeFATE operator includes manifests with components to manage infrastructures composed by multiple clusters, to deploy the cluster itself and to deploy a FL job. Although it is an interesting implementation, it is not compatible with interesting FL frameworks proposed in section 2.5 and its design is not really tailored to the specific needs of hybrid infrastructures composed by cloud and edge components. Therefore, *it cannot satisfy the needs of IoT NGIN MLaaS platform*.

As an alternative, IoT-NGIN will work on a more modular and flexible integration mechanism between the relevant FL frameworks and techniques implemented within IoT-NGIN and Kubeflow as main background technology for the MLaaS platform. In this sense, the integration will be based on the following principles:

[2] https://github.com/kubeflow/community/blob/master/proposals/fate-operator-proposal.md

- Creation of Helm charts for a fast and easy deployment on Kubernetes infrastructure of FL clients.
- Creation of new addons and operators for Kubeflow in order to perform the aggregator role as part of FL workflows.
- Particularisation to work with lightweight versions of Kubernetes like MicroK8s or K3s.

A complete description about the integration will be included in the deliverable D3.3.

4.2 Privacy-preservation in FL within IoT-NGIN

Along with the disruptive appearance of AI technologies and IoT utilization in various domains, data privacy and security became a critical challenge. Federated Learning tends to preserve private data as it does share model parameters rather than the raw data itself. In addition, FL plays a crucial role for these technologies to be employed effectively and productively. Nonetheless, FL needs additional privacy-preserving techniques and robust strategies for finer training, enhancing privacy, and limiting adversarial attacks. In the IoT-NGIN project, the Private Aggregation of Teachers Ensembles (PATE) strategy is utilized for the following reasons:

- It provides strong differential privacy guarantees independent of the learning algorithm.
- It is flexible and adaptive because it is framework-Agnostic, model-agnostic and datatype agnostic.
- The data partitioning scheme that PATE introduces, guarantees that even if a trained model of an FL client has been compromised by a model poisoning attack and the model updates are false, the model can still be correctly aggregated because of the correct updates from the other clients. That being said, PATE introduces the voting system, therefore, if many agree on a value then the few are compromised, or they didn't learn correctly.
- PATE architecture performs also, as an effective legularizer that produces better-behaved models.

PATE is a great strategy to preserve privacy in FL applications. One major shortcoming in PATE is that the *Student* model needs publicly available datasets to be trained, yet sometimes are not available. Therefore, in the next steps of the IoT-NGIN project, additional methods will be explored similar to [124], towards a more advanced PATE scheme.

Within IoT-NGIN, PATE was *re-implemented* in *KERAS 2.0* (TensorFlow2.0+) because the currently available GitHub TensorFlow Federated repository [125] includes a rather old PATE implementation which is developed in TensorFlow 1.0. Therefore, to stay up-to-date and to enable integrations in other solutions within the IoT-NGIN FL flows as well as to facilitate future research work, PATE implementation in KERAS 2.0 has been considered essential.

4.2.1 ML model training using PATE

To test and understand in practice the great advantages of PATE, a sequential ML model for classification was implemented and tested on a Suricata dataset collected from the Synelixis PfSense platform and showed in Figure 26. Suricata is an independent open-source threat detection engine which combines intrusion detection (IDS), intrusion prevention (IPS),

network security monitoring (NSM) and PCAP processing. Some key characteristics are the following:

- High-performance and multi-tasking network intrusion detection system (IDS), prevention (IPS) and security monitoring engine.
- Protects networks, collects, and stores information about any incoming signals.
- Open source - owned by the non-profit foundation Open Information Security Foundation (OISF).
- Suricata attack detector is based on the analysis of signatures and heuristics.

2020-10-29T11:43:24+00:00	suricata[82403]:	[1:2023472:7]	ET	POLICY	External	IP	Lookup	Domain	(myip.opendns	.com
2020-10-29T11:43:24+00:00	suricata[82403]:	[1:2023472:7]	ET	POLICY	External	IP	Lookup	Domain	(myip.opendns	.com
2020-10-29T11:43:24+00:00	suricata[83726]:	[1:2023472:7]	ET	POLICY	External	IP	Lookup	Domain	(myip.opendns	.com
2020-10-29T11:43:24+00:00	suricata[83726]:	[1:2023472:7]	ET	POLICY	External	IP	Lookup	Domain	(myip.opendns	.com
2020-10-29T11:43:55+00:00	suricata[83726]:	[1:2200076:2]	SURICATA	ICMPv4	invalid	checksum	[Classification:	Generic	Protocol	Command
2020-10-29T11:43:55+00:00	suricata[83726]:	[1:2200076:2]	SURICATA	ICMPv4	invalid	checksum	[Classification:	Generic	Protocol	Command
2020-10-29T11:43:55+00:00	suricata[83726]:	[1:2023472:7]	ET	POLICY	External	IP	Lookup	Domain	(myip.opendns	.com
2020-10-29T11:43:55+00:00	suricata[83726]:	[1:2023472:7]	ET	POLICY	External	IP	Lookup	Domain	(myip.opendns	.com
2020-10-29T11:43:55+00:00	suricata[82403]:	[1:2023472:7]	ET	POLICY	External	IP	Lookup	Domain	(myip.opendns	.com
2020-10-29T11:43:55+00:00	suricata[82403]:	[1:2023472:7]	ET	POLICY	External	IP	Lookup	Domain	(myip.opendns	.com
2020-10-29T11:44:05+00:00	suricata[83726]:	[1:2230003:1]	SURICATA	TLS	invalid	handshake	message	[Classification:	Generic	Protocol
2020-10-29T11:44:05+00:00	suricata[83726]:	[1:2230003:1]	SURICATA	TLS	invalid	handshake	message	[Classification:	Generic	Protocol
2020-10-29T11:44:05+00:00	suricata[83726]:	[1:2230010:1]	SURICATA	TLS	invalid	record/traffic	[Classification:	Generic	Protocol	Command
2020-10-29T11:44:05+00:00	suricata[83726]:	[1:2230010:1]	SURICATA	TLS	invalid	record/traffic	[Classification:	Generic	Protocol	Command
2020-10-29T11:44:05+00:00	suricata[83726]:	[1:2230003:1]	SURICATA	TLS	invalid	handshake	message	[Classification:	Generic	Protocol
2020-10-29T11:44:05+00:00	suricata[83726]:	[1:2230003:1]	SURICATA	TLS	invalid	handshake	message	[Classification:	Generic	Protocol
2020-10-29T11:44:05+00:00	suricata[83726]:	[1:2230010:1]	SURICATA	TLS	invalid	record/traffic	[Classification:	Generic	Protocol	Command
2020-10-29T11:44:05+00:00	suricata[83726]:	[1:2230010:1]	SURICATA	TLS	invalid	record/traffic	[Classification:	Generic	Protocol	Command
2020-10-29T11:44:20+00:00	suricata[83726]:	[1:2230003:1]	SURICATA	TLS	invalid	handshake	message	[Classification:	Generic	Protocol
2020-10-29T11:44:20+00:00	suricata[83726]:	[1:2230003:1]	SURICATA	TLS	invalid	handshake	message	[Classification:	Generic	Protocol
2020-10-29T11:44:20+00:00	suricata[83726]:	[1:2230010:1]	SURICATA	TLS	invalid	record/traffic	[Classification:	Generic	Protocol	Command
2020-10-29T11:44:20+00:00	suricata[83726]:	[1:2230010:1]	SURICATA	TLS	invalid	record/traffic	[Classification:	Generic	Protocol	Command
2020-10-29T11:44:20+00:00	suricata[83726]:	[1:2230003:1]	SURICATA	TLS	invalid	handshake	message	[Classification:	Generic	Protocol
2020-10-29T11:44:20+00:00	suricata[83726]:	[1:2230003:1]	SURICATA	TLS	invalid	handshake	message	[Classification:	Generic	Protocol
2020-10-29T11:44:20+00:00	suricata[83726]:	[1:2230010:1]	SURICATA	TLS	invalid	record/traffic	[Classification:	Generic	Protocol	Command
2020-10-29T11:44:20+00:00	suricata[83726]:	[1:2230010:1]	SURICATA	TLS	invalid	record/traffic	[Classification:	Generic	Protocol	Command

Figure 26: Suricata Logs extracted by Synelixis pfSense platform.

A crucial step in the ML cycle is data preprocessing, which includes all the essential techniques to transform the raw data to a format that will facilitate the building and training of ML models. In particular, for intrusion detection datasets, like the Suricata dataset, great attention is needed. Considering this work [126] and empirically, data preprocessing steps for the Suricata dataset are defined as follows:

- Cleaning (Removing punctuations like .,!()$#&*%@, remove URLs, remove stop words).
- Filtering.
- Organization (test must be lower case, etc.).
- Transforming procedures (Tokenization, etc.).

83.235.169.221:11353,208.67.222.222:53,2,et	policy	external	ip	lookup	domain	myip	opendns	com	in	dns	lookup,udp
192.168.168.178:37788,208.67.222.222:53,2,et	policy	external	ip	lookup	domain	myip	opendns	com	in	dns	lookup,udp
192.168.168.238:8,192.168.1.1:0,3,suricata	icmpv	invalid	checksum,icmp								
192.168.168.178:60786,208.67.222.222:53,2,et	policy	external	ip	lookup	domain	myip	opendns	com	in	dns	lookup,udp
83.235.169.221:57953,208.67.222.222:53,2,et	policy	external	ip	lookup	domain	myip	opendns	com	in	dns	lookup,udp
52.114.158.91:443,192.168.168.139:39204,3,suricata	tls	invalid	handshake	message,tcp							
52.114.158.91:443,192.168.168.139:39204,3,suricata	tls	invalid	record	traffic,tcp							
192.168.168.139:39204,52.114.158.91:443,3,suricata	tls	invalid	handshake	message,tcp							
192.168.168.139:39204,52.114.158.91:443,3,suricata	tls	invalid	record	traffic,tcp							
131.253.33.219:443,192.168.168.6:65201,3,suricata	tls	invalid	handshake	message,tcp							
131.253.33.219:443,192.168.168.6:65201,3,suricata	tls	invalid	record	traffic,tcp							
192.168.168.6:65201,131.253.33.219:443,3,suricata	tls	invalid	handshake	message,tcp							
192.168.168.6:65201,131.253.33.219:443,3,suricata	tls	invalid	record	traffic,tcp							
83.235.169.221:33261,208.67.222.222:53,2,et	policy	external	ip	lookup	domain	myip	opendns	com	in	dns	lookup,udp
192.168.168.178:60125,208.67.222.222:53,2,et	policy	external	ip	lookup	domain	myip	opendns	com	in	dns	lookup,udp
13.94.251.244:443,192.168.168.178:59542,3,suricata	tls	invalid	handshake	message,tcp							
13.94.251.244:443,192.168.168.178:59542,3,suricata	tls	invalid	record	traffic,tcp							
13.94.251.244:443,192.168.168.178:59542,3,suricata	tls	invalid	record	type,tcp							
192.168.168.178:59542,13.94.251.244:443,3,suricata	tls	invalid	record	type,tcp							
192.168.168.178:59542,13.94.251.244:443,3,suricata	tls	invalid	handshake	message,tcp							
192.168.168.178:59542,13.94.251.244:443,3,suricata	tls	invalid	record	traffic,tcp							
52.114.128.10:443,192.168.168.178:36978,3,suricata	tls	invalid	handshake	message,tcp							

Figure 27: Data after Preprocessing steps.

In Figure 27, the data after the preprocessing transformation is presented. Suricata tool classifies incoming requests based on a risk factor of being an adversarial attack. The requests take a value from 1 to 3 depending on the risk level. These labels were used as ground truths in order to facilitate the evaluation of our example. To train a classification algorithm a sequential neural network (Figure 28) is designed and developed in KERAS.

```
1.  Model: "sequential"
2.  _________________________________________________________________
3.  Layer (type)                 Output Shape              Param #
4.  =================================================================
5.  dense (Dense)                (None, 512)               10752
6.  _________________________________________________________________
7.  dense_1 (Dense)              (None, 128)               65664
8.  _________________________________________________________________
9.  activation (Activation)      (None, 128)               0
10. _________________________________________________________________
11. dense_2 (Dense)              (None, 64)                8256
12. _________________________________________________________________
13. dense_3 (Dense)              (None, 3)                 195
14. _________________________________________________________________
15. activation_1 (Activation)    (None, 3)                 0
16. =================================================================
```

Figure 28: A simple sequential neural network model was developed, and this is the architecture.

Typically, PATE needs at least 250 teachers to perform well, but due to the limitation of the available dataset the paradigm limits the teachers to 100. In the next steps of IoT-NGIN, additional experiments with PATE using more teachers and different DL/ML networks will be explored.

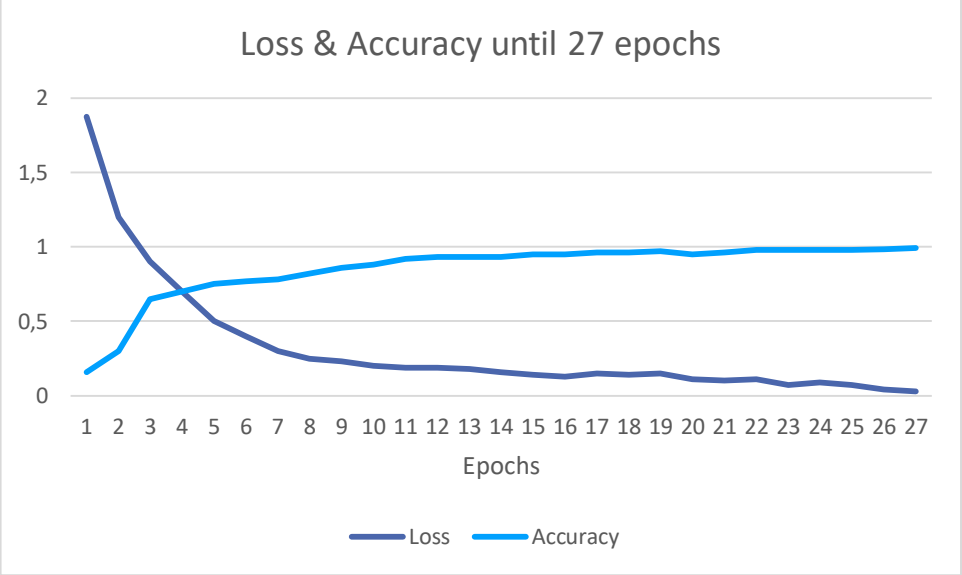


Figure 29: Aggregated values for Teacher models: Loss and Accuracy diagram over 27 epochs, Final Loss: 0.0224 & Accuracy: 0.9922.

The trained Teacher models with PATE resulted in 0.0224 Loss value and 0.9922 Accuracy Figure 29, which are some fine results for such a simple neural network. Training using PATE enhances privacy. In this example, Teachers are trained on private partitioned data and the student on locally collected data, therefore privacy is preserved. In Figure 30, Student model metrics are presented, the final Loss equals to 0.03 and the Accuracy to 0.9912.

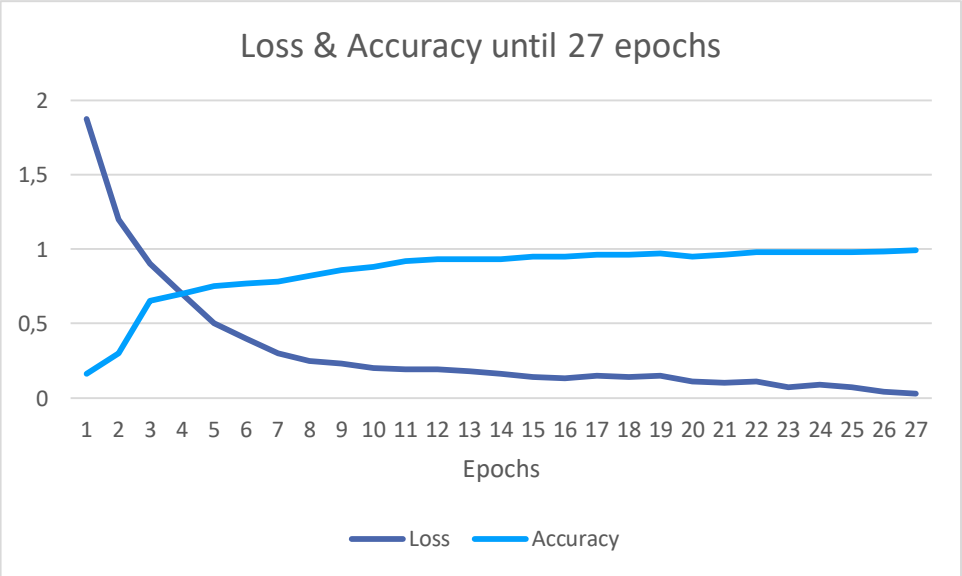
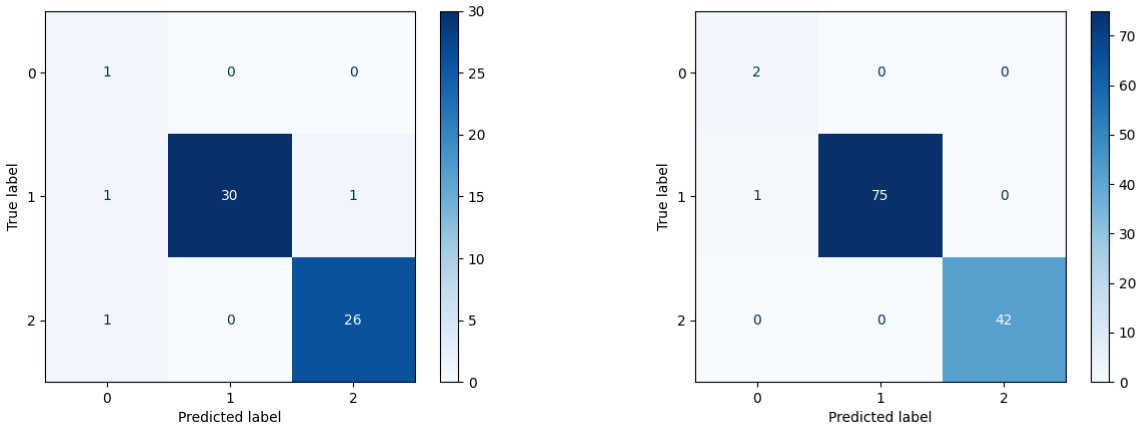


Figure 30: Student model Loss and Accuracy diagram over 27 epochs, final Loss: 0.03 & Accuracy: 0.9912.

A Confusion matrix is a table that is used to define the performance of the classification task. In this example, the confusion matrices in Figure 31 correspond to a Teacher and the Student's model respectively.



a) b)

Figure 31. Confusion matrices that show the predicted labels compared to the true labels. a)This Confusion matrix relates to a Teacher model and b) to the Student's model.

In this task, a simple ML model is trained using PATE as the privacy-preserving mechanism. Although more trivial Differential Privacy mechanisms can be utilized, PATE provides a more sophisticated and effective approach and also, computes the data-dependent differential privacy bounds to estimate the cost of training the student model.

4.2.2 DL model training using Flower

Another experimentation copes with the Flower FL framework. In particular, a DL model for classification on CIFAR10 [127] dataset is trained in a federated manner employing the Flower framework and TensorFlow. CIFAR10 is a commonly used dataset for machine learning training and computer vision algorithms which includes 50000 32x32 colour training images and 10000 test images of 10 different labels (aeroplane, automobile, bird, cat, dog, frog, horse, ship, truck) where each label corresponds to 10000 images. The model used is the MobileNetV2 [128], a convolutional neural network for image classification that is 53 layers deep. In this example, the Adam [129] optimizer is used, a method for stochastic gradient optimization. Regarding the loss function, the *sparse_categorical_crossentropy* is used, which is suitable for multi-class classification model training and the metric chosen for model evaluation is the accuracy of the training. The example trains the model in a federated setup with 10 clients and for 20 epochs.

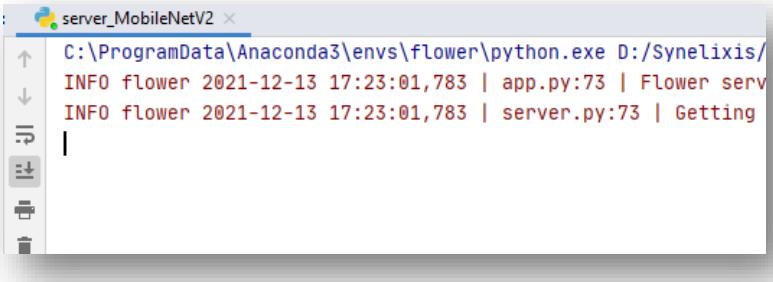


Figure 32: Flower server up and running and waiting for Flower clients to join.

In Figure 32, the Flower's server is up and running as well as waiting for Flower clients to join and collaboratively train the shared model, Figure 33, shows a client that is training the model. Loss is high and accuracy is low because it is the initial step of the learning process.

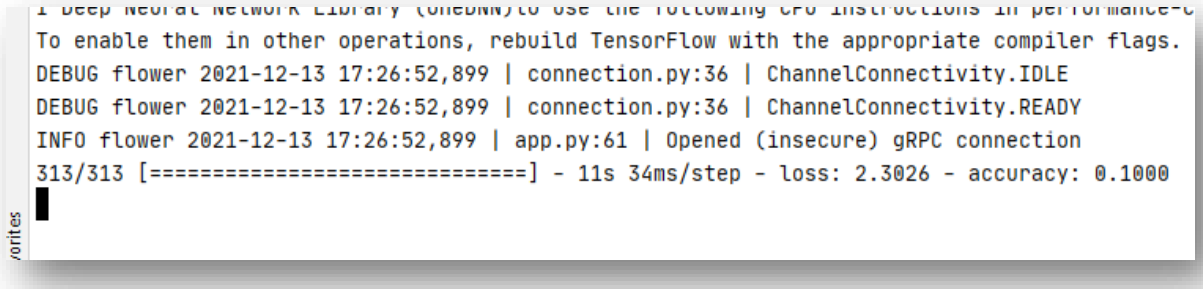


Figure 33: Flower clients training the shared model.

Flower does not provide a privacy-preserving mechanism, to the best of authors' knowledge. As the next step of IoT-NGIN work, PATE will be integrated into Flower as a privacy-preserving mechanism.

4.3 FL Frameworks with Privacy-preserving mechanisms in IoT-NGIN

In the IoT-NGIN project, three FL frameworks (Flower, PySyft + PyGrid and TensorFlow Federated) are selected, each one for different network setup (IoT Node, Edge Node, Cloud Node) as the ones that may be offered via PPFLaaS. Table 9 tabulates the privacy-preserving FL frameworks considered in IoT-NGIN, under the network setup perspective.

Table 9: Privacy-preserving FL frameworks under the network setup perspective.

FL Framework	Network Category (IoT Node, Edge Node, Cloud)	Connectivity Layers	Most suited type of Learning
Flower	IoT Node - Edge Node	All possible layer combinations	Broad support for multiple types of Learning
PySyft + PyGrid	Cloud - Edge Node	IoT Node - Edge Node or Cloud to Edge Node or Cloud to IoT Node	Broad support for multiple types of Learning but it does not initialize GPUs for training
TensorFlow Federated	IoT Node - Edge Node	All possible layer combinations	Broad support for multiple types of Learning

The ML requirements of the LLs use cases have been analyzed in deliverable document D3.1 "Enhancing deep learning / reinforcement learning" [130], identifying the use cases that could employ FL methods in model training. In Table 10, the most suitable privacy-preserving FL methods are identified for the LL use cases, based on the already elicited ML requirements.

Table 10: Mapping privacy-preserving FL tools to IoT-NGIN use cases.

LL	Use Case Description	Flower + PATE	PySyft + PyGrid + PATE or DP	TensorFlow Federated + PATE or DP
Twin Smart Cities	Traffic Flow & Parking Prediction	X		
Twin Smart Cities	Crowd Management	X		
Smart Agriculture	Crop diseases prediction & irrigation precision	X	X	

Industry 4.0	Human–centered safety in a self-aware indoor factory		X	X
Industry 4.0	Digital powertrain and condition monitoring			X
Smart Energy	Move from reacting to acting in smart grid monitoring and control	X		X
Smart Energy	Driver-friendly dispatchable EV charging	X		X

The FL framework exploitation within the IoT-NGIN project shall address the requirements from the LLs. Regarding Smart Cities use cases, the Flower framework is the most suitable because it can easily be deployed in multiple-client scenarios like the ones in smart cities because both UCs integrate numerous IoT devices.

4.3.1 Technical Design

Since Flower does not provide a privacy-preserving mechanism, to the best of authors' knowledge, within IoT-NGIN, PATE will be integrated into Flower as a privacy-preserving mechanism. In this section, the architecture of IoT-NGIN's FL framework (Flower + PATE) is described.

PATE's design consists of one student and many teachers while Flower is based on a *multi-client to a central server* scheme. In this case (Figure 34), data portioning is not needed as in the traditional PATE strategy because each IoT Node collects data independently from the central server, using IoT sensors or smart devices. The only thing that must be considered is the data partition size and the fact that it should be equal. In any case, the training dataset of all clients can be cropped to the minimum training data size which exists among all clients. This is crucial because PATE is designed to aggregate models of the same dimensions.

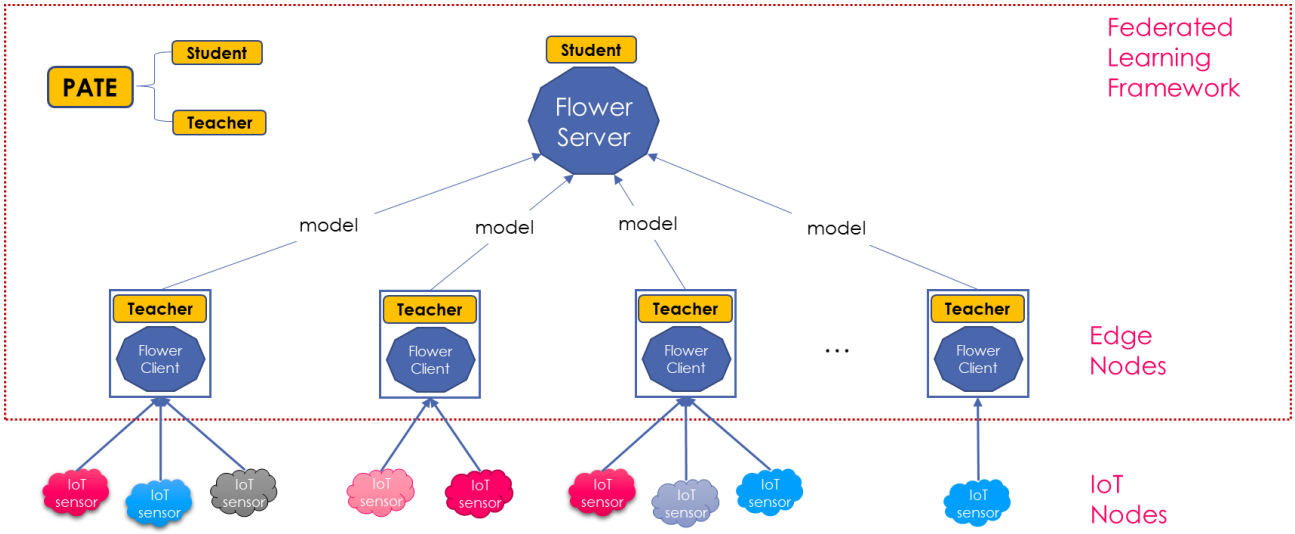


Figure 34: An Overview of IoT-NGINs' FL setup: *Flower* Federated Learning framework with *PATE* as the privacy-preserving mechanism.

Figure 34, shows an overview of the Flower FL framework with PATE being the essential privacy-preserving mechanism, on a system with many participants. In this example, IoT sensors and smart devices are connected to a Flower client and provide their collected data for the training process of the Teacher. In this system, every Flower client includes a teacher model, and the Flower Server includes the Student. In Figure 35, a more detailed view of the Flower setup is given. This representation shows that an orchestrator is added in the pipeline to monitor the whole flow but also, to train the student and to perform Teacher's aggregation.

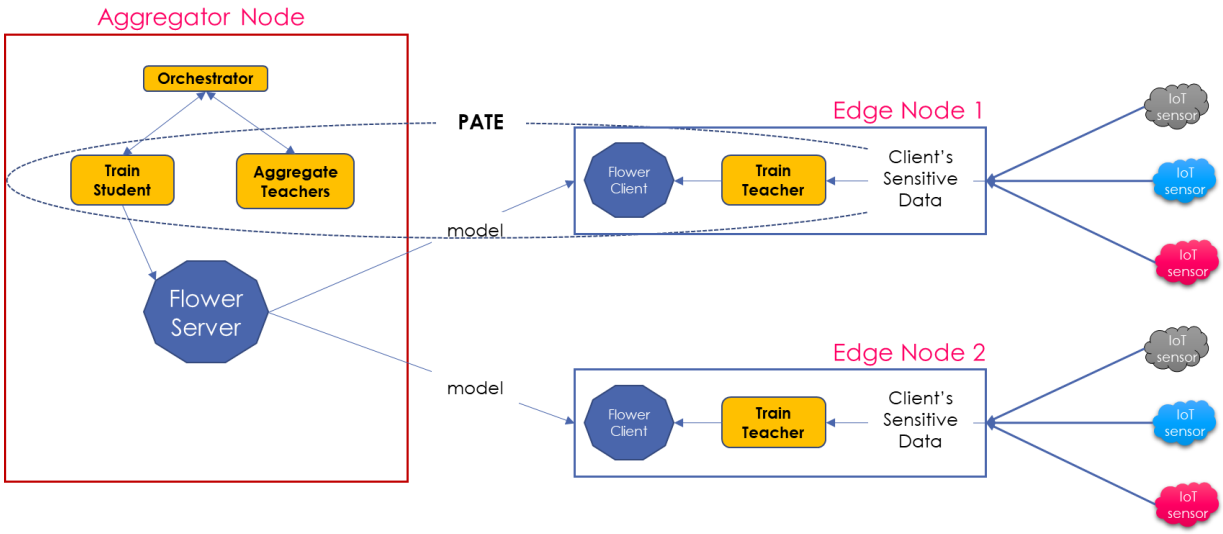


Figure 35: A detailed view of IoT-NGINs' FL setup: *Flower* framework with *PATE* as the privacy-preserving mechanism.

5 Installation Guide

The code and the installation procedures are available on the project Gitlab repository, accessible at:
https://gitlab.com/h2020-iot-ngin/enhancing_iot_cybersecurity_and_data_privacy/privacy-preserving-federated-learning
A brief description is given also here.

Running the PATE example[3] which uses a simple sequential neural network that learns to classify Suricata signals can be executed with the following commands:

```
1. cd pate-keras
2. pip install -r requirements.txt
```

and then run the following command:

```
1. python pate_orchestrator.py -i input_dir -t number_of_teachers -c number_of_classes -data
   data_input -labels label_input
```

Running the Flower example[4] which employs the MobilNetV2 model for classification on the CIFAR10 dataset can be executed with the following commands:

```
1. cd flower-experiment
2. pip install -r requirements.txt
3. python server.py
```

and in 10 different terminals run:

```
1. python client.py
```

[3] https://gitlab.com/h2020-iot-ngin/enhancing_iot_cybersecurity_and_data_privacy/privacy-preserving-federated-learning/privacy-preserving-fl-mechanisms/pate-keras
[4] https://gitlab.com/h2020-iot-ngin/enhancing_iot_cybersecurity_and_data_privacy/privacy-preserving-federated-learning/privacy-preserving-fl-experiments/flower-experiment

6 User Guide

This section describes the processes to run a Flower example with 10 clients. For running a multiple-client scenario in Flower framework to train a DL model for classification using TensorFlow, the following steps are needed.

- First, Flower and TensorFlow must be installed

```
1. pip install flwr
2. pip install tensorflow
```

- Then, the client and the server scripts must be developed

Client script
The following script must be constructed and saved as *client.py*

```
1. import flwr as fl
2. import tensorflow as tf
3.
4. (x_train, y_train), (x_test, y_test) = tf.keras.datasets.cifar10.load_data()
5. model = tf.keras.applications.MobileNetV2((32, 32, 3), classes=10, weights=None)
6. model.compile("adam", "sparse_categorical_crossentropy", metrics=["accuracy"])
7.
8.
9. class CifarClient(fl.client.NumPyClient):
10.     def get_parameters(self):
11.         return model.get_weights()
12.
13.     def fit(self, parameters, config):
14.         model.set_weights(parameters)
15.         model.fit(x_train, y_train, epochs=20, batch_size=64, steps_per_epoch=3)
16.         return model.get_weights(), len(x_train), {}
17.
18.     def evaluate(self, parameters, config):
19.         model.set_weights(parameters)
20.         loss, accuracy = model.evaluate(x_test, y_test)
21.         return loss, len(x_test), {"accuracy": accuracy}
22.
23.
24. fl.client.start_numpy_client("127.0.0.1:8080", client=CifarClient())
25.
```

Server script

The following script must be constructed and saved as *server.py*

```
1. import flwr as fl
2. from flwr.server.strategy import FedAvg
3.
4. # Start Flower server for three rounds of federated learning
5. if __name__ == "__main__":
6.     # FedAvg is the default strategy used when you start the server without a custom strategy
7.     strategy = FedAvg(
8.         # Minimum number of connected clients before sampling e.g. 10
```

```
9.          min_available_clients=10,
10.
11.         # Fraction of clients which should participate in each round
12.         fraction_fit=0.3
13.     )
14.     fl.server.start_server("127.0.0.1:8080", strategy=strategy, config={"num_rounds": 3})
```

7 Conclusions

This report has presented in an extensive form the state-of-the-art in Federated Learning theory, tools and algorithms and reasons why Federated Learning is necessary for privacy-preserving machine learning with many clients on decentralized data.

Moreover, this document presented an extensive comparative analysis over open-source FL tools and through this broad analysis, the most suitable for the scope of IoT-NGIN are indicated for future exploitation.

The design of the IoT-NGIN privacy-preserving FL as a service has been also presented as a generic approach, offering training via different FL frameworks as a service.

The report includes the initial approaches and experimentations of FL tools both available open-sourced and developed within IoT-NGIN. The first version of the privacy-preserving FL tools is presented and evaluated through simple ML models. Moreover, the enhancement of Flower with the PATE privacy-preserving technique has been designed and specified.

During the next steps of IoT-NGIN activities towards the design, development and evaluation of privacy-preserving FL Frameworks the following tasks will be explored, addressed and finalized:

- Implementation of the privacy analysis methodology for the PATE framework to compute the privacy guarantees of the student model.
- Finalization of integration of Flower with PATE.
- Exploration of different ML models with Flower, PySyft + PyGrid and TensorFlow Federated on heterogeneous environments.

The final version of the privacy-preserving FL framework of IoT-NGIN will be presented in D3.3 "Enhanced IoT federated deep learning/ reinforcement ML", due at the end of the third quarter of 2022.

8 References

[1] "federated-learning-collaborative @ ai.googleblog.com." https://ai.googleblog.com/2017/04/federated-learning-collaborative.html

[2] H. Brendan McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas, "Communication-efficient learning of deep networks from decentralized data," *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017*, vol. 54, 2017.

[3] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated Learning: Strategies for Improving Communication Efficiency," pp. 1–10, 2016, [Online]. Available: http://arxiv.org/abs/1610.05492

[4] J. Kone˘, H. B. Mcmahan, D. Ramage, and P. Richt, "Federated Optimization : Distributed Machine Learning for On-Device Intelligence," pp. 1–38, 2016.

[5] B. M. D. Ramage, "Blog Federated Learning : Collaborative Machine Learning," pp. 1–5, 2021, [Online]. Available: https://ai.googleblog.com/2017/04/federated-learning-collaborative.html

[6] F. Directions and V. Smith, "Federated Learning: Challenges, Methods, and Future Directions," pp. 1–21.

[7] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated Machine Learning : Concept and Applications," vol. 10, no. 2, pp. 1–19, 2019.

[8] L. U. Khan, W. Saad, Z. Han, E. Hossain, and C. S. Hong, "Federated learning for internet of things: Recent advances, taxonomy, and open challenges," *IEEE Communications Surveys and Tutorials*, vol. 23, no. 3, pp. 1759–1799, 2021, doi: 10.1109/COMST.2021.3090430.

[9] O. A. Wahab, A. Mourad, H. Otrok, and T. Taleb, "Federated Machine Learning: Survey, Multi-Level Classification, Desirable Criteria and Future Directions in Communication and Networking Systems," *IEEE Communications Surveys and Tutorials*, vol. 23, no. 2, pp. 1342–1397, 2021, doi: 10.1109/COMST.2021.3058573.

[10] E. Jeong, S. Oh, H. Kim, J. Park, M. Bennis, and S. Kim, "Communication-Efficient On-Device Machine Learning: Federated Distillation and Augmentation under Non-IID Private Data," no. Nips, 2018, [Online]. Available: https://arxiv.org/abs/1811.11479

[11] N. Agarwal, A. T. Suresh, F. Yu, S. Kumar, and H. Brendan McMahan, "CPSGD: Communication-efficient and differentially-private distributed SGD," *Advances in Neural Information Processing Systems*, vol. 2018-Decem, pp. 7564–7575, 2018, [Online]. Available: https://arxiv.org/pdf/1805.10559.pdf

[12] F. Sattler, S. Wiedemann, K. R. Muller, and W. Samek, "Robust and Communication-Efficient Federated Learning from Non-i.i.d. Data," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 9, pp. 3400–3413, 2020, doi: 10.1109/TNNLS.2019.2944481.

[13] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, and M. Chen, "In-edge AI: Intelligentizing mobile edge computing, caching and communication by federated learning," *IEEE Network*, vol. 33, no. 5, pp. 156–165, 2019, doi: 10.1109/MNET.2019.1800286.

[14] H. T. Nguyen, N. Cong Luong, J. Zhao, C. Yuen, and D. Niyato, "Resource Allocation in Mobility-Aware Federated Learning Networks: A Deep Reinforcement Learning Approach," *IEEE World Forum on Internet of Things, WF-IoT 2020 - Symposium Proceedings*, 2020, doi: 10.1109/WF-IoT48130.2020.9221089.

[15] S. Wang *et al.*, "Adaptive Federated Learning in Resource Constrained Edge Computing Systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019, doi: 10.1109/JSAC.2019.2904348.

[16] W. Wu, L. He, W. Lin, R. Mao, C. Maple, and S. Jarvis, "SAFA: A Semi-Asynchronous Protocol for Fast Federated Learning with Low Overhead," *IEEE Transactions on Computers*, vol. 70, no. 5, pp. 655–668, 2021, doi: 10.1109/TC.2020.2994391.

[17] T. T. Anh, N. C. Luong, D. Niyato, D. I. Kim, and L. C. Wang, "Efficient training management for mobile crowd-machine learning: A deep reinforcement learning approach," *IEEE Wireless Communications Letters*, vol. 8, no. 5, pp. 1345–1348, 2019, doi: 10.1109/LWC.2019.2917133.

[18] D. Li and J. Wang, "FedMD: Heterogenous Federated Learning via Model Distillation," no. NeurIPS, pp. 1–8, 2019, [Online]. Available: http://arxiv.org/abs/1910.03581

[19] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing Federated Learning on Non-IID Data with Reinforcement Learning," *Proceedings - IEEE INFOCOM*, vol. 2020-July, pp. 1698–1707, 2020, doi: 10.1109/INFOCOM41043.2020.9155494.

[20] Y. Ruan, X. Zhang, S.-C. Liang, and C. Joe-Wong, "Towards Flexible Device Participation in Federated Learning," vol. 130, 2020, [Online]. Available: http://arxiv.org/abs/2006.06954

[21] A. Koskela and A. Honkela, "Learning Rate Adaptation for Federated and Differentially Private Learning," pp. 1–17, 2018, [Online]. Available: http://arxiv.org/abs/1809.03832

[22] A. Triastcyn and B. Faltings, "Federated Generative Privacy," *IEEE Intelligent Systems*, vol. 35, no. 4, pp. 50–57, 2020, doi: 10.1109/MIS.2020.2993966.

[23] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," *36th International Conference on Machine Learning, ICML 2019*, vol. 2019-June, pp. 1012–1021, 2019, [Online]. Available: https://arxiv.org/pdf/1811.12470.pdf

[24] Y. Han and X. Zhang, "Robust Federated Training via Collaborative Machine Teaching using Trusted Instances," 2019, [Online]. Available: http://arxiv.org/abs/1905.02941

[25] D. Peterson, P. Kanani, and V. J. Marathe, "Private Federated Learning with Domain Adaptation," no. 1, pp. 1–6, 2019, [Online]. Available: http://arxiv.org/abs/1912.06733

[26] S. Masoudnia and R. Ebrahimpour, "Mixture of experts: A literature survey," *Artificial Intelligence Review*, vol. 42, no. 2, pp. 275–293, 2014, doi: 10.1007/s10462-012-9338-y.

[27] H. Brendan McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017*, 2017, vol. 54.

[28] T. Nishio and R. Yonetani, "Client Selection for Federated Learning with Heterogeneous Resources in Mobile Edge," in *IEEE International Conference on Communications*, 2019, vol. 2019-May. doi: 10.1109/ICC.2019.8761315.

[29] X. Yao, C. Huang, and L. Sun, "Two-Stream Federated Learning: Reduce the Communication Costs," *VCIP 2018 - IEEE International Conference on Visual Communications and Image Processing*, pp. 2–5, 2018, doi: 10.1109/VCIP.2018.8698609.

[30] A. Gretton *et al.*, "Optimal kernel choice for large-scale two-sample tests," *Advances in Neural Information Processing Systems*, vol. 2, pp. 1205–1213, 2012, [Online]. Available: http://www.stat.cmu.edu/~siva/Papers/MMD12.pdf

[31] P. Kairouz *et al.*, "Advances and open problems in federated learning," *Foundations and Trends in Machine Learning*, vol. 14, no. 1–2. pp. 1–210, 2021. doi: 10.1561/2200000083.

[32] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology*, vol. 10, no. 2, pp. 1–19, 2019, doi: 10.1145/3298981.

[33] M. Aledhari, R. Razzak, R. M. Parizi, and F. Saeed, "Federated Learning: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Access*, vol. 8. pp. 140699–140725, 2020. doi: 10.1109/ACCESS.2020.3013541.

[34] "Cross-Device Federated learning." [Online]. Available: https://medium.com/accenture-the-dock/instilling-responsible-and-reliable-ai-development-with-federated-learning-d23c366c5efd

[35] D. Gao, C. Ju, X. Wei, Y. Liu, T. Chen, and Q. Yang, "HHHFL: Hierarchical Heterogeneous Horizontal Federated Learning for Electroencephalography," pp. 1–7, 2019, [Online]. Available: http://arxiv.org/abs/1909.05784

[36] J. Zhao, X. Zhu, J. Wang, and J. Xiao, "Efficient client contribution evaluation for horizontal federated learning," *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 2021-June, pp. 3060–3064, 2021, doi: 10.1109/ICASSP39728.2021.9413377.

[37] C. Feng, B. Liu, K. Yu, S. K. Goudos, and S. Wan, "Blockchain-Empowered Decentralized Horizontal Federated Learning for 5G-Enabled UAVs," *IEEE Transactions on Industrial Informatics*, vol. 3203, no. c, pp. 1–1, 2021, doi: 10.1109/tii.2021.3116132.

[38] Y. Wu, S. Cai, X. Xiao, G. Chen, and B. C. Ooi, "Privacy-preserving vertical federated learning for tree-based models," *Proceedings of the VLDB Endowment*, vol. 13, no. 11, pp. 2090–2103, 2020, doi: 10.14778/3407790.3407811.

[39] K. Yang, T. Fan, T. Chen, Y. Shi, and Q. Yang, "A Quasi-Newton Method Based Vertical Federated Learning Framework for Logistic Regression," vol. 2019, 2019, [Online]. Available: http://arxiv.org/abs/1912.00513

[40] N. Angelou *et al.*, "Asymmetric Private Set Intersection with Applications to Contact Tracing and Private Vertical Federated Machine Learning," no. ii, pp. 1–10, 2020, [Online]. Available: http://arxiv.org/abs/2011.09350

[41] D. Romanini *et al.*, "PyVertical: A Vertical Federated Learning Framework for Multi-headed SplitNN," pp. 1–9, 2021, [Online]. Available: http://arxiv.org/abs/2104.00489

[42] S. Feng and H. Yu, "Multi-Participant Multi-Class Vertical Federated Learning," 2020, [Online]. Available: http://arxiv.org/abs/2001.11154

[43] J. Sun *et al.*, "Vertical Federated Learning without Revealing Intersection Membership," 2021, [Online]. Available: http://arxiv.org/abs/2106.05508

[44] S. Yang, B. Ren, X. Zhou, and L. Liu, "Parallel Distributed Logistic Regression for Vertical Federated Learning without Third-Party Coordinator," 2019, [Online]. Available: http://arxiv.org/abs/1911.09824

[45] Y. Liu, Y. Kang, C. Xing, T. Chen, and Q. Yang, "A Secure Federated Transfer Learning Framework," *IEEE Intelligent Systems*, vol. 35, no. 4, pp. 70–82, 2020, doi: 10.1109/MIS.2020.2988525.

[46] Y. Chen, X. Qin, J. Wang, C. Yu, and W. Gao, "FedHealth: A Federated Transfer Learning Framework for Wearable Healthcare," *IEEE Intelligent Systems*, vol. 35, no. 4, pp. 83–93, 2020, doi: 10.1109/MIS.2020.2988604.

[47] C. Ju, D. Gao, R. Mane, B. Tan, Y. Liu, and C. Guan, "Federated Transfer Learning for EEG Signal Classification," *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*, vol. 2020-July, no. Dl, pp. 3040–3045, 2020, doi: 10.1109/EMBC44109.2020.9175344.

[48] A. Hard *et al.*, "Federated Learning for Mobile Keyboard Prediction," Nov. 2018, [Online]. Available: http://arxiv.org/abs/1811.03604

[49] M. Chen, R. Mathews, T. Ouyang, and F. Beaufays, "Federated Learning Of Out-Of-Vocabulary Words," pp. 1–6, 2019, [Online]. Available: http://arxiv.org/abs/1903.10635

[50] D. Leroy, A. Coucke, T. Lavril, T. Gisselbrecht, and J. Dureau, "Federated Learning for Keyword Spotting," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2019, vol. 2019-May, pp. 6341–6345. doi: 10.1109/ICASSP.2019.8683546.

[51] T. Yang *et al.*, "Applied Federated Learning: Improving Google Keyboard Query Suggestions," 2018, [Online]. Available: http://arxiv.org/abs/1812.02903

[52] S. Ramaswamy, R. Mathews, K. Rao, and F. Beaufays, "Federated Learning for Emoji Prediction in a Mobile Keyboard," Jun. 2019, [Online]. Available: http://arxiv.org/abs/1906.04329

[53] U. M. Aïvodji, S. Gambs, and A. Martin, "IOTFLA : AA secured and privacy-preserving smart home architecture implementing federated learning," *Proceedings - 2019 IEEE Symposium on Security and Privacy Workshops, SPW 2019*, pp. 175–180, 2019, doi: 10.1109/SPW.2019.00041.

[54] T. Yu *et al.*, "Learning context-aware policies from multiple smart homes via federated multi-task learning," *Proceedings - 5th ACM/IEEE Conference on Internet of Things Design and Implementation, IoTDI 2020*, pp. 104–115, 2020, doi: 10.1109/IoTDI49375.2020.00017.

[55] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, and S. Choudhary, "Federated Learning with Personalization Layers," no. Ml, 2019, [Online]. Available: http://arxiv.org/abs/1912.00818

[56] B. Hu, Y. Gao, L. Liu, and H. Ma, "Federated Region-Learning: An Edge Computing Based Framework for Urban Environment Sensing," *2018 IEEE Global Communications Conference, GLOBECOM 2018 - Proceedings*, pp. 1–7, 2018, doi: 10.1109/GLOCOM.2018.8647649.

[57] N. I. Mowla, N. H. Tran, I. Doh, and K. Chae, "Federated Learning-Based Cognitive Detection of Jamming Attack in Flying Ad-Hoc Network," *IEEE Access*, vol. 8, pp. 4338–4350, 2020, doi: 10.1109/ACCESS.2019.2962873.

[58] K. Bonawitz *et al.*, "TensorFlow Federated: Machine Learning on Decentralized Data." 2020. [Online]. Available: https://www.tensorflow.org/federated

[59] T. Ryffel *et al.*, "A generic framework for privacy-preserving deep learning," pp. 1–5, 2018, [Online]. Available: http://arxiv.org/abs/1811.04017

[60] "FederatedAI/FATE: An Industrial Grade Federated Learning Framework." https://github.com/FederatedAI/FATE

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

[62] T. Ryffel *et al.*, "Federated Learning and Differential Privacy: Software tools analysis, the Sherpa.ai FL framework and methodological guidelines for preserving data privacy," Nov. 2020, [Online]. Available: https://arxiv.org/pdf/2007.00914.pdf

[63] C. He *et al.*, "FedML: A Research Library and Benchmark for Federated Machine Learning," 2020, [Online]. Available: http://arxiv.org/abs/2007.13518

[64] "PFL:Paddle Federated Learning." [Online]. Available: https://paddlefl.readthedocs.io/en/latest/

[65] S. Caldas *et al.*, "LEAF: A Benchmark for Federated Settings," no. NeurIPS, pp. 1–9, 2018, [Online]. Available: http://arxiv.org/abs/1812.01097

[66] G. A. Reina *et al.*, "OpenFL: An open-source framework for Federated Learning," pp. 1–22, 2021, [Online]. Available: http://arxiv.org/abs/2105.06413

[67] I. Kholod *et al.*, "Open-source federated learning frameworks for IoT: A comparative review and analysis," *Sensors (Switzerland)*, vol. 21, no. 1, pp. 1–22, 2021, doi: 10.3390/s21010167.

[68] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the Convergence of FedAvg on Non-IID Data," no. 2019, pp. 1–26, 2019, [Online]. Available: http://arxiv.org/abs/1907.02189

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

[70] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani, "FedPAQ: A Communication-Efficient Federated Learning Method with Periodic Averaging and Quantization," no. 2, 2019, [Online]. Available: http://arxiv.org/abs/1909.13014

[71] 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, pp. 479–489, 2021, doi: 10.1109/jsait.2021.3054610.

[72] L. Liu, J. Zhang, S. H. Song, and K. B. Letaief, "Client-Edge-Cloud Hierarchical Federated Learning," *IEEE International Conference on Communications*, vol. 2020-June, 2020, doi: 10.1109/ICC40277.2020.9148862.

[73] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni, "Federated Learning with Matched Averaging," pp. 1–16, 2020, [Online]. Available: http://arxiv.org/abs/2002.06440

[74] X. Yao, T. Huang, R.-X. Zhang, R. Li, and L. Sun, "Federated Learning with Unbiased Gradient Aggregation and Controllable Meta Updating," pp. 1–16, 2019, [Online]. Available: http://arxiv.org/abs/1910.08234

[75] S. Ek, F. Portet, P. Lalanda, and G. Vega, "A Federated Learning Aggregation Algorithm for Pervasive Computing: Evaluation and Comparison," *2021 IEEE International Conference on Pervasive Computing and Communications, PerCom 2021*, 2021, doi: 10.1109/PERCOM50583.2021.9439129.

[76] M. Mohri, G. Sivek, and A. T. Suresh, "Agnostic federated learning," *36th International Conference on Machine Learning, ICML 2019*, vol. 2019-June, pp. 8114–8124, 2019, [Online]. Available: https://arxiv.org/pdf/1902.00146.pdf

[77] K. Sreenivasan and H. Wang, "Attack of the Tails : Yes , You Really Can Backdoor Federated Learning," pp. 1–28.

[78] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-pruning: Defending against backdooring attacks on deep neural networks," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2018, vol. 11050 LNCS, pp. 273–294. doi: 10.1007/978-3-030-00470-5_13.

[79] J. Steinhardt, P. W. Koh, and P. Liang, "Certified defenses for data poisoning attacks," in *Advances in Neural Information Processing Systems*, 2017, vol. 2017-Decem, no. i, pp. 3518–3530.

[80] S. Wang, X. Wang, S. Ye, P. Zhao, and X. Lin, "Defending DNN adversarial attacks with pruning and logits augmentation," *2018 IEEE Global Conference on Signal and Information Processing, GlobalSIP 2018 - Proceedings*, pp. 1144–1148, 2019, doi: 10.1109/GlobalSIP.2018.8646578.

[81] Z. Sun, P. Kairouz, A. T. Suresh, and H. B. McMahan, "Can You Really Backdoor Federated Learning?," 2019, [Online]. Available: http://arxiv.org/abs/1911.07963

[82] E. Bagdasaryan and C. Tech, "How To Backdoor Federated Learning," vol. 108, 2020.

[83] I. Goodfellow *et al.*, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020, doi: 10.1145/3422622.

[84] X. Zhang and X. Luo, "Exploiting Defenses against GAN-Based Feature Inference Attacks in Federated Learning," 2020, [Online]. Available: http://arxiv.org/abs/2004.12571

[85] S. Taheri, A. Khormali, M. Salem, and J. S. Yuan, "Developing a robust defensive system against adversarial examples using generative adversarial networks," *Big Data and Cognitive Computing*, vol. 4, no. 2, pp. 1–15, 2020, doi: 10.3390/bdcc4020011.

[86] N. Carlini, C. Liu, Ú. Erlingsson, J. Kos, and D. Song, "The secret Sharer: Evaluating and testing unintended memorization in neural networks," in *Proceedings of the 28th USENIX Security Symposium*, 2019, pp. 267–284.

[87] M. Abadi *et al.*, "Deep learning with differential privacy," in *Proceedings of the ACM Conference on Computer and Communications Security*, 2016, vol. 24-28-Octo, no. Ccs, pp. 308–318. doi: 10.1145/2976749.2978318.

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

[89] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Foundations and Trends in Theoretical Computer Science*, vol. 9, no. 3–4, pp. 211–487, 2013, doi: 10.1561/0400000042.

[90] C. Zhao *et al.*, "Secure Multi-Party Computation: Theory, practice and applications," *Information Sciences*, vol. 476, pp. 357–372, 2019, doi: 10.1016/j.ins.2018.10.024.

[91] C. Lefebvre, "On data Banks and Privacy Homomorphisms," *Journal of Pidgin and Creole Languages*, vol. 15, no. 2, pp. 313–337, Dec. 2000, doi: 10.1075/jpcl.15.2.04lef.

[92] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-Preserving Deep Learning via Additively Homomorphic Encryption," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 5, pp. 1333–1345, 2018, doi: 10.1109/TIFS.2017.2787987.

[93] C. Gentry, "A Fully Homomorphic Encryption Scheme," *Dissertation*, no. September, p. 169, 2009, [Online]. Available: http://cs.au.dk/~stm/local-cache/gentry-thesis.pdf

[94] S. Truex *et al.*, "A hybrid approach to privacy-preserving federated learning," in *Proceedings of the ACM Conference on Computer and Communications Security*, 2019, pp. 1–11. doi: 10.1145/3338501.3357370.

[95] R. Xu, N. Baracaldo, Y. Zhou, A. Anwar, and H. Ludwig, "HybridAlpha : An Efficient Approach for Privacy-Preserving Federated Learning".

[96] C. Wang, J. Liang, M. Huang, B. Bai, K. Bai, and H. Li, "Hybrid Differentially Private Federated Learning on Vertically Partitioned Data," 2020, [Online]. Available: http://arxiv.org/abs/2009.02763

[97] Y. Li, Y. Zhou, A. Jolfaei, D. Yu, G. Xu, and X. Zheng, "Privacy-Preserving Federated Learning Framework Based on Chained Secure Multiparty Computing," *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6178–6186, 2021, doi: 10.1109/JIOT.2020.3022911.

[98] V. Mugunthan, R. Rahman, and L. Kagal, "BlockFLow: An Accountable and Privacy-Preserving Solution for Federated Learning," 2020, [Online]. Available: http://arxiv.org/abs/2007.03856

[99] P. C. Mahawaga Arachchige, P. Bertok, I. Khalil, D. Liu, S. Camtepe, and M. Atiquzzaman, "Local Differential Privacy for Deep Learning," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 5827–5842, 2020, doi: 10.1109/JIOT.2019.2952146.

[100] Y. Zhao *et al.*, "Local Differential Privacy-Based Federated Learning for Internet of Things," *IEEE Internet of Things Journal*, vol. 8, no. 11, pp. 8836–8853, 2021, doi: 10.1109/JIOT.2020.3037194.

[101] M. Seif, R. Tandon, and M. Li, "Wireless Federated Learning with Local Differential Privacy," *IEEE International Symposium on Information Theory - Proceedings*, vol. 2020-June, pp. 2604–2609, 2020, doi: 10.1109/ISIT44484.2020.9174426.

[102] J. Lee and C. Clifton, "How much is enough? Choosing ε for differential privacy," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7001 LNCS, pp. 325–340, 2011, doi: 10.1007/978-3-642-24861-0_22.

[103] J. Xu, W. Zhang, and F. Wang, "A(DP)^2SGD: Asynchronous Decentralized Parallel Stochastic Gradient Descent with Differential Privacy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021, doi: 10.1109/TPAMI.2021.3107796.

[104] 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

[105] C. Baek, S. Kim, D. Nam, and J. Park, "Enhancing Differential Privacy for Federated Learning at Scale," *IEEE Access*, vol. 9, pp. 148090–148103, 2021, doi: 10.1109/access.2021.3124020.

[106] C. Crane, "What is Homomorphic Encryption?," *Hashed Out*, 2019. https://blog.openmined.org/what-is-homomorphic-encryption/

[107] L. J. M. Aslett, P. M. Esperança, and C. C. Holmes, "A review of homomorphic encryption and software tools for encrypted statistical machine learning," pp. 1–21, 2015, [Online]. Available: http://arxiv.org/abs/1508.06574

[108] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, and Y. Liu, "BatchCrypt: Efficient homomorphic encryption for cross-silo federated learning," in *Proceedings of the 2020 USENIX Annual Technical Conference, ATC 2020*, 2020, pp. 493–506.

[109] N. Dowlin, R. Gilad-Bachrach, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," in *33rd International Conference on Machine Learning, ICML 2016*, 2016, vol. 1, pp. 342–351.

[110] F. Bourse, M. Minelli, M. Minihold, and P. Paillier, "Fast homomorphic evaluation of deep discretized neural networks," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2018, vol. 10993 LNCS, pp. 483–512. doi: 10.1007/978-3-319-96878-0_17.

[111] E. Chou, J. Beal, D. Levy, S. Yeung, A. Haque, and L. Fei-fei, "Faster CryptoNets : Leveraging Sparsity for Real-World Encrypted Inference".

[112] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan, "GAZELLE: A low latency framework for secure neural network inference," in *Proceedings of the 27th USENIX Security Symposium*, 2018, pp. 1651–1668.

[113] O. Goldreich and Y. Oren, "Definitions and properties of zero-knowledge proof systems," *Journal of Cryptology*, vol. 7, no. 1, pp. 1–32, 1994, doi: 10.1007/BF00195207.

[114] P. Mohassel and Y. Zhang, "SecureML: A System for Scalable Privacy-Preserving Machine Learning," *Proceedings - IEEE Symposium on Security and Privacy*, pp. 19–38, 2017, doi: 10.1109/SP.2017.12.

[115] N. Kilbertus, A. Gascón, M. Kusner, M. Veale, K. P. Gummadi, and A. Weiler, "Blind justice: Fairness with encrypted sensitive attributes," *35th International Conference on Machine Learning, ICML 2018*, vol. 6, pp. 4123–4137, 2018.

[116] N. Papernot, M. Abadi, Ú. Erlingsson, I. Goodfellow, and K. Talwar, "Semi-supervised Knowledge Transfer for Deep Learning from Private Training Data," *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, Oct. 2016, [Online]. Available: http://arxiv.org/abs/1610.05755

[117] 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 - Conference Track Proceedings*, 2018, pp. 1–34.

[118] C. Dwork, G. N. Rothblum, and S. Vadhan, "Boosting and differential privacy," *Proceedings - Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pp. 51–60, 2010, doi: 10.1109/FOCS.2010.12.

[119] I. Mironov, "Rényi Differential Privacy," *Proceedings - IEEE Computer Security Foundations Symposium*, pp. 263–275, 2017, doi: 10.1109/CSF.2017.11.

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

[121] C. He, M. Annavaram, and S. Avestimehr, "Towards Non-I.I.D. and Invisible Data with FedNAS: Federated Deep Learning via Neural Architecture Search," pp. 1–6, 2020, [Online]. Available: http://arxiv.org/abs/2004.08546

[122] C. Thapa, M. A. P. Chamikara, S. Camtepe, and L. Sun, "SplitFed: When Federated Learning Meets Split Learning," 2020, [Online]. Available: http://arxiv.org/abs/2004.12088

[123] K. Bonawitz *et al.*, "Practical Secure Aggregation for Federated Learning on User-Held Data," no. Nips, 2016, [Online]. Available: http://arxiv.org/abs/1611.04482

[124] S. Yovine, F. Mayr, S. Sosa, and R. Visca, "An Assessment of the Application of Private Aggregation of Ensemble Models to Sensible Data," *Machine Learning and Knowledge Extraction*, vol. 3, no. 4, pp. 788–801, 2021, doi: 10.3390/make3040039.

[125] N. Papernot, "PATE implementation in Tensorflow @github.com." [Online]. Available: https://github.com/tensorflow/privacy/tree/master/research/pate_2017

[126] T. Ahmad and M. N. Aziz, "Data preprocessing and feature selection for machine learning intrusion detection systems," *ICIC Express Letters*, vol. 13, no. 2, pp. 93–101, 2019, doi: 10.24507/icicel.13.02.93.

[127] Alex Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," *Asha*, vol. 34, no. 4, 2009, [Online]. Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

[128] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, 2018, doi: 10.1109/CVPR.2018.00474.

[129] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pp. 1–15, 2015, [Online]. Available: https://arxiv.org/pdf/1412.6980.pdf

[130] IoT-NGIN, "Enhancing deep learning / reinforcement learning," pp. 1–48, 2021.