University of Zurich^UZH

MASTER THESIS – Communication Systems Group, Prof. Dr. Burkhard Stiller

Quantifying the Trustworthiness Level of Federated Learning Models

Ning Xie
Zürich, Switzerland
Student ID: 20-736-104

Supervisor: Dr. Alberto Huertas Celdran & Muriel Franco
Date of Submission: November 9, 2022

University of Zurich
Department of Informatics (IFI)
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland

ifi

Master Thesis
Communication Systems Group (CSG)
Department of Informatics (IFI)
University of Zurich
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland
URL: http://www.csg.uzh.ch/

Abstract

In the last decade, the rise of Deep Learning (DL) in the development of Artificial Intelligence (AI) has greatly improved the performance of AI models which are becoming increasingly relevant as a support to the human decision-making process. With the ever widening spread of AI applications powered on Big Data, centralized machine learning became challenging due to the existing data silos in many industries where data contain sensitive information. The rising concern for data privacy in AI is promoting the development of privacy-preserving Machine and Deep Learning (ML/DL) techniques such as Federated Learning (FL) where model training is performed collaboratively by distributed data contributors in a decentralized manner. FL enables data privacy by design since local data are not exposed. The increasing interest and adoption of FL systems prompt the need to investigate the ability to trust the decisions made by FL models as compared to centralized machine learning. There is a large body of existing literature on the topic of Trustworthy AI where the requirements are drawn out for an AI system under the five pillars of trust: i) robustness, ii) privacy, iii) fairness, iv) explainability and v) accountability. These pillars were developed in the context of traditional ML/DL systems. As the attention of AI shifts to FL, more efforts are needed to identify trustworthiness pillars and evaluation metrics relevant for FL models. This work analyzed the existing requirements for trustworthiness evaluation in AI and adapted the pillars and metrics for state-of-the-art FL models. A comprehensive taxonomy for Trustworthy FL is proposed as a result of the analysis. Based on the taxonomy, an evaluation algorithm, *FederatedTrust*, was designed and implemented as a third-party Python library which can be imported as a plugin to an FL development framework to evaluate the trustworthiness level of FL models. The *FederatedTrust* library harnesses the meta data and configuration settings of FL models gathered from the development framework and generates inputs and outputs for trustworthiness analysis based on the metrics identified in the taxonomy. At the end of an FL training, a report containing the trust scores of each metric and pillar that make up the aggregated trustworthiness level is generated for the FL model created. The report helps to identify the areas impacting trust within the model configuration and execution so that improvements can be made to make the model more trustworthy. Validation of the algorithm was conducted in the form of experiments to test the usefulness of the trustworthiness report generated by *FederatedTrust* under different FL settings. Observations and discussions were made on the experiment results to analyze what can be improved in the future development of this evaluation framework for Trustworthy FL.

Zusammenfassung

In den letzten zehn Jahren hat der Aufstieg von Deep Learning (DL) in der Entwicklung der künstlichen Intelligenz (KI) die Leistung von KI-Modellen erheblich verbessert, die als Unterstützung für den menschlichen Entscheidungsprozess immer relevanter werden. Mit der immer größeren Verbreitung von KI-Anwendungen, die auf Big Data basieren, wurde zentralisiertes maschinelles Lernen aufgrund der bestehenden Datensilos in vielen Branchen, in denen Daten sensible Informationen enthalten, zu einer Herausforderung. Die zunehmende Sorge um den Datenschutz in der KI fördert die Entwicklung von Techniken zum maschinellen und tiefen Lernen (ML/DL), bei denen die Privatsphäre gewahrt wird, wie z. FL ermöglicht Datenschutz durch Design, da lokale Daten nicht offengelegt werden. Das zunehmende Interesse und die zunehmende Akzeptanz von FL-Systemen erfordern die Untersuchung der Fähigkeit, den von FL-Modellen getroffenen Entscheidungen im Vergleich zum zentralisierten maschinellen Lernen zu vertrauen. Es gibt eine große Menge an vorhandener Literatur zum Thema vertrauenswürdige KI, in der die Anforderungen an ein KI-System unter den fünf Säulen der Verantwortlichkeit des Vertrauens dargelegt werden: i) Robustheit, ii) Datenschutz, iii) Fairness, iv) Erklärbarkeit und v) . Diese Säulen wurden im Kontext traditioneller ML/DL-Systeme entwickelt. Da sich die Aufmerksamkeit der KI auf FL verlagert, sind weitere Anstrengungen erforderlich, um Vertrauenswürdigkeitssäulen und Bewertungsmetriken zu identifizieren, die für FL-Modelle relevant sind. Diese Arbeit analysierte die bestehenden Anforderungen an die Vertrauenswürdigkeitsbewertung in der KI und passte die Säulen und Metriken für State-of-the-Art-FL-Modelle an. Eine umfassende Taxonomie für Trustworthy FL wird vorgeschlagen ein Ergebnis der Analyse. Basierend auf der Taxonomie wurde ein Bewertungsalgorithmus, FederatedTrust, entworfen und als Drittanbieter-Python-Bibliothek implementiert, die als Plugin in ein FL-Entwicklungsframework importiert werden kann, um die Vertrauenswürdigkeit von FL-Modellen zu bewerten. Die FederatedTrust-Bibliothek nutzt die Metadaten und Konfigurationseinstellungen von FL-Modellen, die aus dem Entwicklungsframework gesammelt wurden, und generiert Ein- und Ausgaben für die Vertrauenswürdigkeitsanalyse basierend auf den in der Taxonomie identifizierten Metriken. Am Ende eines FL-Trainings wird für das erstellte FL-Modell ein Bericht erstellt, der die Vertrauenswerte jeder Metrik und Säule enthält, die das aggregierte Vertrauenswürdigkeitsniveau bilden. Der Bericht hilft bei der Identifizierung der Bereiche, die sich auf das Vertrauen innerhalb der Modellkonfiguration und -ausführung auswirken, sodass Verbesserungen vorgenommen werden können, um das Modell vertrauenswürdiger zu machen. Die Validierung des Algorithmus wurde in Form von Experimenten durchgeführt, um die Nützlichkeit des von FederatedTrust generierten Vertrauenswürdigkeitsberichts unter verschiedenen FL-Einstellungen zu testen. Es wurden Beobachtungen und Diskussionen zu den Versuchsergebnissen durchgeführt, um zu analysieren, was in der zukünftigen Entwicklung dieses Bewertungsrahmens für Trustworthy FL verbessert werden kann.

Acknowledgments

I would like to express my sincere appreciation for the Communication Systems Group at the University of Zurich for providing me with this opportunity to conduct an insightful thesis project. Many thanks to Dr. Alberto Huertas Celdran, Dr. Muriel Franco and Prof. Dr. Burkhard Stiller for supervising and guiding me throughout the process. A special thanks goes to Dr. Alberto Huertas Celdran and Mr. Pedro Miguel Sánchez Sánchez for the continuous encouraging support and feedback during the project.

I would also like to thank the FederatedScope project team who made such a user friendly and yet powerful federated learning deployment framework where students and researchers can easily test out this new machine learning paradigm. The timely project provided me with many inspirations and help for the deployment and validation of the evaluation algorithm.

Lastly, I am happy that this project came to a good conclusion with meaningful contributions after six months of hard work. I am very grateful for my family and partner for their unconditional support during this challenging period of time.

Contents

Abstract **i**

Acknowledgments **iv**

1 Introduction **1**

1.1 Motivation 1

1.2 Description of Work 3

1.3 Thesis Outline 4

2 Related Work **5**

2.1 Trustworthy AI 5

2.1.1 Related Scientific Work 6

2.1.2 Related Tools 7

2.2 State-of-the-Art FL 8

2.3 Trustworthy FL 9

2.3.1 Related Scientific Work 9

2.3.2 Related Tools 11

3 Six Pillars of Trust in FL **13**

3.1 Robustness Pillar 14

3.1.1 Resilience to Attacks 14

3.1.2 System-level Robustness 16

3.1.3 Algorithm-level Robustness 17

3.1.4 Client and Data Reliability 18
3.1.5 Limitations 19
3.2 Privacy Pillar 19
3.2.1 Privacy-preserving Approaches 20
3.2.2 Information Gain/Loss 21
3.2.3 Uncertainty 21
3.2.4 Indistinguishability 22
3.2.5 Limitations 23
3.3 Fairness Pillar 24
3.3.1 Client Selection Fairness 24
3.3.2 Performance Fairness 25
3.3.3 Group-level Fairness 25
3.3.4 Class Distribution 26
3.3.5 Limitations 27
3.4 Explainability Pillar 28
3.4.1 Interpretability 28
3.4.2 Post-hoc Explainability Methods 29
3.4.3 Limitations 30
3.5 Accountability Pillar 31
3.5.1 FactSheet Completeness 31
3.5.2 Monitoring 33
3.5.3 Limitations 33
3.6 Architectural Soundness Pillar 34
3.6.1 Client Management 34
3.6.2 Model Management 35
3.6.3 Optimization 36
3.6.4 Limitations 36
3.7 Trustworthy FL Taxonomy 37

4 Trustworthy FL Evaluation Algorithm **38**

4.1 Algorithm Design . . . 39

4.1.1 Exploring Open-Source FL Frameworks . . . 39

4.1.2 FederatedScope . . . 41

4.1.3 Context and Assumptions . . . 42

4.1.4 Requirements and Constraints . . . 42

4.1.5 Architecture . . . 43

4.2 Implementation . . . 46

4.2.1 Taxonomy . . . 46

4.2.2 Metric Definitions . . . 47

4.2.3 FederatedTrust Algorithm . . . 50

4.3 Deployment . . . 63

4.3.1 Use Case . . . 64

4.3.2 Result Discussion . . . 66

4.4 Limitations . . . 71

5 Summary and Conclusions **72**

5.1 Future Work . . . 73

References **73**

Abbreviations **82**

Glossary **84**

List of Figures **85**

List of Tables **86**

A Metric Calculation References **88**

Chapter 1

Introduction

1.1 Motivation

The last decade was a revolutionary time for the development of AI with the rise of DL [1]. Started from IBM Watson [2], ImageNet [3], AlphaGo [4] and Siri [5], all the way to GPT-3 [6], DALL·E 2 [7] and Tesla Autopilot [8], AI can now see, speak, paint and drive like a human. Traditionally, the hype was focused on achieving ever higher accuracy and performance of the algorithms. However, since AI models became more complicated with the use of black box algorithms such as Deep Neural Network (DNN) and Convolutional Neural Network (CNN) [9], explainable AI (XAI) was needed to promote understandability and trust in AI. In parallel, the growth of Big Data through the widespread of AI-powered applications in our daily life and society has made it possible for data owners to access and influence consumer behaviours, public opinions and individual decisions. Meanwhile, we start to hear more mishaps of AI in the news as more responsibilities got delegated to AI such as in criminal risk profiling [10], childcare support fraud detection [11] and autonomous driving [12]. This lead to a rising concern on data privacy and the general safety of AI applications.

Since 2016, laws and regulations have been drawn in response to the growing need for improved data protection and privacy, such as the General Data Protection Regulation (GDPR) within the European Union (EU) and the California Consumer Privacy Act (CCPA) in the state of California. On the AI technology front, a revolutionary step was the development of Federated Learning (FL), a type of decentralized machine learning with the preservation of data privacy. It was originally born out of the need to build models efficiently while keeping sensitive user data distributed on mobile devices by Google in 2016 [13]. In a way, FL is the solution to the data silo and fragmentation problems caused by the new legislation that prohibits freely sharing of data and forces data to be maintained by isolated data owners [14]. Over the past few years the research community of FL has grown and the global FL market size is projected to be USD 210 million by 2028 [15]. In fact, the federated approach is regarded as a key player in the future of digital health where clinical data can remain confidential [16] and in the emerging AI market where there is growing need for scale, inter-organizational collaboration and data integrity.

Beyond data privacy and safety, there is a general push from the society and the government for Responsible AI (RAI) [17] systems. Trustworthy AI is an emerging concept towards RAI that encompasses several of the existing terminologies such as XAI, ethical AI [18], fair AI [19] and so on. In 2019, the ethics guidelines for AI set up by the European Commission defined three high-level guidelines and four principles of trustworthiness that an AI should follow [20]. A tremendous amount of research and discussions also went into Trustworthy AI in recent years [21][22][23][24][25][26] that produced various interpretations and definitions of trust in the context of AI systems.

To better understand the trustworthiness of an FL model, comparisons to classical centralized learning and extension on existing Trustworthy AI studies have to be made. Similar to the traditional way of centralized ML/DL, FL is subject to the common risks of algorithmic bias, adversarial attacks, data privacy breaches and reliability issues. However, unlike centralized learning, FL involves different and more stakeholders, actors, information exchanges, communication infrastructures and attack surfaces. It also presents new challenges in architectural design, opportunities for privacy-preserving standards, new perspectives on fairness and explainability beyond the underlying ML/DL models. [27]

In order to make FL models trustworthy, a methodology to evaluate and quantify trustworthiness in such systems needs to be developed. It requires examining the applicability of existing evaluation methods from Trustworthy AI as well as curating new trustworthiness pillars and metrics specifically for the context of FL that are missing from the current literature. In the existing literature, the development on the technicality of trustworthy algorithms overshadows the development on the evaluation frameworks for FL. The goal of this thesis is to bridge the gap by studying and extending the Trustworthy AI taxonomy with requirements specifically for Trustworthy FL and to implement an evaluation algorithm based on the taxonomy. The main contributions of the thesis are the building of the taxonomy and the implementation of the prototype evaluation algorithm for trustworthiness level of FL models.

1.2 Description of Work

This work surveyed the state-of-the-art FL frameworks and the requirements for trust in AI models. Each pillar and metric used to evaluate the trustworthiness of classical and federated ML/DL models was reviewed, compared and eventually curated into a general taxonomy for trust in FL models. Six pillars have been identified as the main building blocks of the taxonomy: i) robustness, ii) privacy, iii) fairness, iv) explainability, v) accountability and vi) architectural soundness. Each pillar represents a dimension of trust that is further broken down into different notions which are then quantified by a number of metrics.

Based on the taxonomy, methods of metric calculation were surveyed and investigated strictly following the distributed nature of FL systems and the priority of preserving data privacy. A list of feasible metrics and their theoretical calculation methods were collected.

As the goal was to implement a prototype trustworthiness quantification algorithm that could be deployed on an FL system, various state-of-the-art open-source FL development and simulation frameworks, including TensorFlow Federated (TFF) [28], Flower [29], FLUTE [30], LEAF [31] and FederatedScope [32], were surveyed and compared for their compatibility to the project. Eventually, FederatedScope, a recent project by the Data Analytics and Intelligence Lab (DAIL) of Alibaba DAMO Academy [33], was chosen as the reference framework to deploy the prototype algorithm for demonstration. The FederatedScope framework uses the most up-to-date technologies and provides a comprehensive environment to simulate various types of FL models with flexible configurations and attack simulation capabilities.

Considering the inner workings of the FederatedScope framework, the prototype algorithm was designed and implemented as a Python library/package that could be imported into the framework to generate inputs and outputs needed for the trust evaluation process. The Python package, named *FederatedTrust* [34], works by harnessing the FL model configurations, the built-in evaluation pipeline and the messaging channels of the framework.

For the validation, four experiments with various numbers of clients and training rounds were executed for the use case of training an image classifier on the FEMNIST [31] dataset in a cross-device client-server based FL context using a baseline FederatedAveraging (FedAvg) [13] algorithm. After each experiment run, the trustworthiness evaluation results were compiled into a JSON report directly under the experiment's evaluation output directory together with the other performance evaluation results. The goal was to compare the experiment results to identify the factors impacting the trustworthiness level in different scenarios of federation and to assess the usefulness of the *FederatedTrust* algorithm.

1.3 Thesis Outline

This thesis contains four main chapters. Chapter 1 introduces the recent development of AI and the background of trustworthy AI and FL that motivate the topic. It outlines the goal, the main contributions and the methodology of the thesis work.

Chapter 2 contains findings from the literature review on the related work in Trustworthy AI and the state-of-the-art FL. In addition, existing work on the topic of Trustworthy FL from the perspectives of privacy, fairness, robustness and design were surveyed as a foundation for the taxonomy.

Chapter 3 presents the first contribution of this work with the detailed analysis of six Trustworthy FL pillars: robustness, privacy, fairness, explainability, accountability and architectural soundness. In the analysis of each pillar, the defining notions and metrics are explained and limitations in terms of feasibility and effectiveness are discussed. This chapter concludes with the comprehensive taxonomy on Trustworthy FL that was created from the analysis.

Chapter 4 presents the second contribution of this work with the detailed explanation and discussions on the design and implementation of the prototype trustworthiness evaluation algorithm, *FederatedTrust*. The algorithm design section includes the main design process, decisions, requirements and architecture. The implementation section contains the list of metric definitions and the code structure of the algorithm. The deployment section demonstrates the use of *FederatedTrust* with one specific use case in an FL framework and presents results from the experiment runs. Observations on the results are discussed.

Chapter 5 concludes this work by providing summary and conclusions. Possible limitations and future work on the topic are discussed.

Chapter 2

Related Work

As mentioned in the introduction, a large body of literature on Trustworthy AI has emerged in recent years along side with the growing interest in FL research. This section details the findings from the literature review of Trustworthy AI (2.1), State-of-the-Art FL (2.2) and Trustworthy FL (2.3). Each section concludes with examples of the relevant tools that have been developed.

2.1 Trustworthy AI

Over the past few years, researchers, society and governing bodies have proposed various definitions and interpretations of Trustworthy AI from multiple perspectives. In 2019, the High-Level Expert Group on Artificial Intelligence (AI HLEG) of the European Commission [20] defined three foundations of Trustworthy AI: i) lawful, ii) ethical and iii) robust. Under the three foundations, there are four principles: i) respect for human autonomy, ii) prevention of harm, iii) fairness and iv) explicability. Following the principles there are seven key requirements for the realisation of Trustworthy AI: i) human agency and oversight, ii) technical robustness and safety, iii) privacy and data governance, iv) transparency, v) diversity, non-discrimination and fairness, vi) societal and environmental wellbeing and vii) accountability.

At the same time, AI researchers also developed specific approaches and techniques that an AI system should adopt to be trustworthy. The systematic reviews on Trustworthy AI conducted by Liu et al. [21] and Li et al. [22] summarized five key pillars of trust: robustness, privacy, fairness, explainability and accountability. The existing work related to these five areas in AI are presented in the following section.

2.1.1 Related Scientific Work

Robustness

The research on robustness in AI mainly focuses on adversarial robustness. Carlini et al. [35] provided a comprehensive survey of methodologies on evaluating adversarial robustness. They discussed ways to define adversarial goals and capabilities and provided a guideline for the best practices. Liu et al. [21] provided the taxonomy of different types of threat models in AI and discussed the representative defense methods for evasion and poisoning attacks. Li et al. [22] discussed robustness in terms of generazaility to diverse data distributions, algorithmic vulnerabilities and system-level security.

Privacy

Privacy is a big topic in machine learning. Al-Rubaie and Chang [36] discussed the potential privacy threats for ML applications regarding the current methods of collecting data and building ML models. Their work touched on privacy attacks such as reconstruction attacks, model inversion attacks, membership inference attacks and de-anonymization. They also presented cryptographic approaches such as homomorphic encryption and perturbation approaches such as differential privacy as privacy-preserving ML techniques. Liu et al. [37] provided a comprehensive taxonomy of privacy attacks in ML and discussed using ML models to help protect privacy. A survey by Wagner and Eckhoff [38] explained and discussed about over eighty privacy metrics and developed a method to help identify the right privacy metrics for a given scenario by asking nine questions. A study by Mehner et al. [39] generalized a differential privacy model to a more understandable interpretation of epsilon for measuring a global privacy risk metric.

Fairness

Feuerriegel et al. [19] introduced and explained the term "fair AI" based on the origins and mathematical notions. They further discussed the sources of unfairness in AI and the algorithms for achieving fair AI. Pessach and Shmueli [40] conducted a comprehensive review on fairness in ML and detailed the different causes of unfairness and the various measurements of algorithmic bias such as disparate impact, demographic parity, equalized odds and equal opportunity. They also discussed the trade-offs between different fairness measurements and between fairness and accuracy.

Explainability

Doshi-Velez and Kim [41] provided the definition for interpretability in AI and a taxonomy for evaluating interpretability. They proposed a data-driven approach to discover facotrs of interpretability. The work by Arrieta et al. [42] is a comprehensive review on the topic of XAI and provides a taxonomy of transparency. They categorized ML models by their level of transparency and explained the post-hoc explainability methods for non-transparent models. In the area of post-hoc explainability methods, Ribeiro et al. [43] proposed the Local Interpretable Model-Agnostic Explanations (LIME) method to explain the predictions of any machine learning classifier by learning a linear Support Vector Machine (SVM). Blanco-Justicia et al. [44] proposed using shallow decision trees to explain the behaviour of deep learning models. In another research, Lundberg and Lee [45] presented a unified framework for interpreting predictions, Shapley Additive exPlanations (SHAP), which assigns each feature an importance value for a particular prediction.

Accountability

Wieringa [46] defined algorithmic accountability as a networked account for a socio-technical algorithmic system following the various stages of the system's lifecycle. During the lifecycle, multiple actors (i.e. stakeholders, developers, users) have the obligation to explain and justify their use, design and/or decisions concerning the system and the subsequent effects of that conduct. Li et al. [22] suggested using checklist-based assessments to evaluate accountability. Raji et al. [47] proposed an auditing framework for the development and deployment of large-scale artificial intelligence systems by learning lessons from aerospace, medical services and finance sectors. Arnold et al. [48] introduced AI FactSheet which is modeled after a supplier's declaration of conformity (SDoC) to show a product conforms to a standard or technical regulation. Their work envisioned such document to contain the purpose, performance, safety, security, and provenance information which would be completed by AI service providers and examined by consumers to promote transparency and trust.

2.1.2 Related Tools

The previous UZH master project work [49] implemented a trust certification algorithm for traditional ML/DL models combining four pillars of trust - fairness, explainability, robustness and training methodology. The algorithm was deployed on a web application to allow users to calculate the trustworthiness of their models by uploading the model parameters and the dataset used.

The IBM AI Factsheet 360, based on the FactSheet project [48], is a platform with a list of example templates illustrating how to properly use a FactSheet for various machine learning projects.

The IBM AI 360 Toolkit [50] is platform with a collection of three tools for exploring three pillars of trust in AI - fairness, explainability and robustness. The AI Fairness 360 is an extensible open source toolkit for examining, reporting and mitigating discrimination and bias in machine learning models throughout the AI application lifecycle. The AI Explainability 360 contains eight state-of-the-art algorithms for interpretable machine learning as well as metrics for explainability. The Adversarial Robustness 360 Toolbox (ART) is a Python Library for machine learning security that enables developers and researchers to defend and evaluate ML/DL models and applications against the adversarial threats of evasion, poisoning, extraction, and inference attacks.

2.2 State-of-the-Art FL

Yang et al. [51] defined an FL system as a learning process where two or more parties or data owners collaboratively train a model during which any data owner does not expose its data to others. Based on the partitioning of the data among the parties, Yang et al. [51] further categorized the learning process into Horizontal Federated Learning (HFL) where datasets share the same feature space but different in samples and Vertical Federated Learning (VFL) where datasets share overlapping sample ID space but differ in feature space. Federated Transfer Learning (FTL) applies to the scenarios that datasets do not share any overlapping sample or feature spaces. In other terms, Kairouz et al. [27] categorized FL settings into "cross-device" and "cross-silo" respectively. Cross-device setting involves mobile devices as clients and is by default HFL, while cross-silo is usually among multiple organizations and can be either HFL or VFL depending on whether the organizations share sample space or feature space.

The work by Lo et al. [52] outlines a typical architecture of FL which involves the roles of a learning coordinator (i.e. system owner or manager) and data contributors (i.e. local model trainers). Yin et al. [53] explain that in HFL there are two main communication architectures: client-server and peer-to-peer. Client-server, also known as centralized FL, is where N clients collaboratively train a model with the help of a server. Peer-to-peer, also known as decentralized FL is when there is no central server and clients communicate model updates to one another. As for VFL, the two common communication architectures are an architecture with a third-party coordinator and one without. Li et al. [54] further explain that in cross-device setting the manager is usually a powerful central server, as in the case of client-server architecture, while in cross-silo setting the manager can be one dominating organization among the participants, as in the scenario without a third-party coordinator.

An FL system can be viewed as a large-scale distributed system with numerous participants and different components, therefore designing such a system requires software system design thinking apart from the machine learning knowledge. Lo et al. [52] proposed several recommended design patterns based on a systematic literature review which covers the areas of client and model management. A well maintained client management system enables the tracking of dishonest or dropout nodes and improves the performance of the model with optimized client selection. A good model management system ensures traceability of local model contributions and improves accountability and communication efficiency.

Regarding how the training process works, the computation happens on both the manager and the participants. A most widely used basic framework for cross-device HFL is the FedAvg algorithm proposed by McMahan et al. [13] when they first introduced the term Federated Learning in 2016. In FedAvg, the central server first broadcasts an initial global model and training parameters to selected parties before training starts. Then in each iteration, each party receives the current global model and updates it with their local training data before sending the updated models back to the central server. Next, the server aggregates all the received local models into a new global model. The process repeats for a number of iterations and the global model of the server is the final output.

For VFL, the work by Yang et al. [51] illustrates that a typical process involves a third-party collaborator and multiple participating organizations where one holds all the label data. The first part of the process is encrypted entity alignment where encryption-based user ID alignment techniques are used to align entities across the data from different organizations. The second part of the process is encrypted model training where parties encrypt and exchange intermediate gradient results with the help of the collaborator.

In an earlier study in 2018, Nilsson et al. [55] compared the performance of FedAvg with Federated Stochastic Variance Reduced Gradient (FSVRG) [56] and CO-OP [57] and reported that FedAvg achieved the highest accuracy. Since then, researchers have proposed several improvements to FedAvg. Reddi et al. [58] proposed the FedOpt algorithm that incorporates adaptivity in FL by allowing the server to update weights with a server optimizer on top of simply averaging the collected weights. Li et al. [59] proposed FedProx that tackles heterogeneity in federated networks by updating the model with a proximal regularizer and provides convergence guarantees when learning over data from Non-IID distribution. Li et al. [60] proposed another approach called FedBN that tackles the features shift Non-IID issue by using local batch normalization before averaging models. T Dinh et al. [61] proposed pFedMe, an effective personalized FL approach to address data heterogeneity, in which the personalized model and global model are decoupled with Moreau envelops.

Given the above background knowledge of state-of-the-art FL, important works related to each Trustworthy AI pillar in the context of FL models were surveyed and presented in the following sections.

2.3 Trustworthy FL

2.3.1 Related Scientific Work

Robustness

Although FL already provides a first level of privacy protection by not sharing local training data, the framework is still vulnerable to privacy attacks in certain cases. Jere et al. [62] provided a taxonomy of attacks on FL that included two categories of attacks: model performance attacks and data privacy attacks. Model performance attacks can be done by data or model poisoning where the integrity of the training data or gradient updates are compromised. Data privacy attacks include membership inference attacks, model inversion attacks and Generative Adversarial Networks (GAN) reconstruction attacks where adversaries can gain information on sensitive properties. Bhagoji et al. [63] reported that even a highly constrained adversary could carry out stealthy model poisoning attacks highlighting the vulnerability of an FL setting. Jere et al. [62] suggested using Differential Privacy (DP), robust aggregation and outlier detection as main defenses based on a survey. Naseri et al. [64] showed empirical evidence that DP could defend against backdoor attacks and mitigate white-box membership inference attacks in FL. Muñoz-González et al. [65] introduced Adaptive Federated Averaging (AFA). AFA is a Byzantine-robust

FL algorithm that detects and discards bad or malicious client updates at every iteration by comparing the similarity of these individual updates to the one for the aggregated model. Rodríguez-Barroso et al. [66] proposed Robust Filtering of one-dimensional Outliers (RFOut-1d), a new FL approach resilient to model-poisoning backdoor attacks by filtering out the model updates of the adversarial clients. In another study on robustness, Alfarra et al. [67] revealed that a simple federated averaging technique was effective in building more certifiably-robust models.

Privacy

As seen from the types of vulnerabilities and attacks on FL, privacy is the central point of focus since parameter gradients are shared among participants and the server. There has been a lot of studies on Privacy-preserving Federated Learning (PPFL) techniques. Three main categories are: i) encryption-based, ii) perturbation-based and iii) anonymization-based. In the encryption-based category, Dong et al. [68] designed a privacy-preserving protocol against semi-honest adversary by combining TernGrad with secret sharing and homomorphic encryption. Hardy et al. [69] developed a privacy-preserving entity resolution and an additively homomorphic encryption scheme for VFL setting. Bonawitz et al. [70] designed the Secure Aggregation aggregator by leveraging Secure Multiparty Computation (SMC) to compute sums of model parameter updates from individual users' devices in a secure manner. In the pertubation-based category, the global DP scheme has been widely used in many FL methods. Choudhury et al. [71] demonstrated that global DP offered strong level of privacy in protecting sensitive health data in a FL framework. Geyer et al. [72] proposed a procedure using DP to ensure that a learned model does not reveal whether a client participated during decentralized training. Unlike perturbation-based techniques, anonymization-based techniques can provide privacy defense without compromising data utility. Choudhury et al. [73] proposed using the k-anonymity scheme at the client level together with a global anonymization mapping process to achieve better privacy preservation and model performance than DP-based FL models. For measuring privacy in FL, Liu et al. [74] proposed a novel method to approximate the mutual information between local gradient updates and batched input data during each round of training.

Fairness

Fairness is another point of focus for FL as multiple parties are contributing data to the model training and are eventually rewarded with the same aggregated global model. A survey by Shi et al. [75] provide an overview of fairness notions adopted in fairness-aware FL approaches. The notions include accuracy parity which measures the degree of uniformity in performance across FL client devices, selection fairness which aims to mitigate bias and reduce under representation and never representation, and contribution fairness which aims to distribute payoff proportionately to the contributions of clients. Yue et al. [76] proposed GIFAIR-FL, a framework that imposed group and individual fairness to FL settings, by penalizing the spread in the loss of clients to drive the optimizer to fair solutions. Zhang et al. [77] developed FairFL that facilitated fairness across all demographic groups by employing a Multi-agent Reinforcement Learning (MARL) based scheme to solve the fair classification problem in FL by enforcing an optimal client selection policy on each client. Huang et al. [78] proposed a long-term fairness constraint that took into consideration an expected guaranteed chosen rate of clients that the selection scheme must fulfill. Fan et al. [79] proposed the completed federated Shapley value (ComFedSV) to evaluate data owners' contributions in FL based on solving a low-rank matrix completion problem for the utility matrix.

Explainability

Even with active research on XAI, there are still challenges specifically to FL models as most client data are private and cannot be read or analyzed. For VFL, some explainability methods such as feature importance can reveal underlying feature information from other parties. For an HFL model, since clients share the same feature space, Wang [80] suggested that it can be safely explained by calculating the Shapley value of each feature using the definition by Molnar [81]. But for VFL models, their work proposed a variant version of SHAP [45] by combining the parties' features into individual united feature space so that parties do not get information of the features from other parties. Chen et al. [82] proposed EVFL, a credible federated counterfactual explanation method [83] to evaluate feature importance for VFL models by utilizing the Kullback-Leibler (KL) divergence [84] to minimize the distribution of the counterfactual and query instances in the client party.

Accountability

Even though FL models are promising regarding privacy, they would require far more transparency and trustworthiness as compared to classical centralized ML/DL models. Baracaldo et al. [85] from IBM Research introduced the Accountable FL FactSheet Framework (AFˆ2 Framework) that could instrument accountability in FL models by fusing verifiable claims with tamper-evident facts. The framework requires different actors (i.e. project owner, data owner, aggregator) to log claims about the various processes during the lifecycle in a distributed immutable ledger. They also expanded the AI FactSheet project to account for the complex model compositions of FL. Desai et al. [86], Mugunthan et al. [87] and Awan et al. [88] incorporated smart contracts to introduce different auditing mechanisms to FL models by leveraging the immutability and decentralized trust properties of blockchain.

2.3.2 Related Tools

Caldas et al. [31] developed LEAF, na open-source benchmarking framework for FL, based on three metrics - performance, amount of computing resources and weighted accuracy. LEAF provides a suite of federated datasets for experiment and produces granular statistical and system analysis.

Chai et al. [89] developed FedEval, an open-source evaluation framework for FL systems, based on five metrics - accuracy, communication, time efficiency, privacy and robustness (ACTPR). FedEval was designed as a benchmarking system with a built-in evaluation model. For accuracy, the evaluation strategy is to compare the federated learning accuracy with the centralized training accuracy to determine whether the FL model has achieved better or worse accuracy. The communication metric measurement relies on the number of communication rounds and the total amount of data transmission during training. The time efficiency metric measures the overall time needed for getting a converged model and the time needed for sub modules in the model. The privacy metric is calculated by implementing two state-of-the-art gradient attacks in the framework and check the model accuracy or review attack results. The robustness metrics is simply the performance under Non-IID data.

The IBM AI FactSheet 360 platform [90] provides an example FactSheet template for Accountable Federated Learning.

Chapter 3

Six Pillars of Trust in FL

Despite the active research and development to make AI trustworthy, there is a lack of systematic surveys or studies on how to improve the trustworthiness of FL models as a whole. In thoery, the five key pillars are applied to FL as well. However, in order to make meaningful and effective evaluations in the FL setting, where privacy and security are priorities, adaptations and additions of notions, metrics and evaluation methods are needed.

Based on the literature review of state-of-the-art FL, this work proposes a new pillar, Architectural Soundness Pillar, in order to capture the complex compositions and design challenges of FL systems in the development of Trustworthy FL.

In the following sections, the first major contribution of this thesis is presented. The six pillars of trust in FL - robustness (3.1), privacy (3.2), fairness (3.3), explainability (3.4), accountability (3.5), architectural soundness (3.6) - are introduced and explained, together with their underlying notions and evaluation metrics, followed by discussion of the limitations.

If not specified otherwise, FL means horizontal FL with baseline FedAvg as this is the most widely used model and the main use case in this work.

3.1 Robustness Pillar

Robustness is regarded as one of the three foundations of trustworthy AI, along with lawfulness and ethics, as defined by the AI HLEG of the European Commission [20]. AI systems must be technically robust to ensure that they are not open to malicious use or bring harm to humans. A robust AI system should be able to withstand input perturbations, erroneous inputs, unseen data or distributional shifts [22]. In practice, this means resilience to attacks, system robustness, high level of accuracy and reliable results. Building resilience in an AI system requires assessing potential forms of attacks, employing defense mechanisms and verifying the system behaviour under unexpected situations. To achieve system level robustness, it requires assessing the system functionalities and operational reliability. High level of accuracy is an indication of performance that falls under algorithm level robustness which also considers generalizability. Lastly, this work proposes adding client and data reliability as a notion for robustness in FL because reliable client resources increases the probability of successful training, and quality data are the key to any reliable ML/DL models.

3.1.1 Resilience to Attacks

According to the literature, FL models are susceptible to poisoning attacks during training and privacy attacks during the learning process. Poisoning attacks in FL have two major categories: data poisoning and model poisoning. In data poisoning attacks, typically the integrity of the training data is compromised so as to compromise the performance of the model. Common methods are flipping or permuting the labels and inserting backdoor patterns or perturbations to the training data. Model poisoning attacks have a broader range and the goal is to manipulate the training procedure. In FL, this could be gradient manipulation, or model update poisoning attack, which is performed by corrupting the updates of a client directly or during model exchanges. Defence against poisoning attacks is therefore a desirable property of a resilient FL model [91]. Privacy attacks include class representative inference attack which is performed by a GAN attack and membership inference attack which is done by passively observing updated model parameters and performing inference. The current state-of-the-art defense against privacy attacks heavily relies on differential privacy and secure multiparty computation, which are discussed in the privacy pillar (3.2).

Evaluation of this notion is performed by first checking if the FL model is equipped with any desirable defense mechanisms and then verifying the model's defense capabilities against representative attacks empirically. For demonstrating dense capabilities, there are the concepts of empirical and certified robustness. Empirical robustness demonstrates how well a model performs against known adversaries by computing the minimal perturbation that the attacker must introduce for a successful attack. Certified robustness goes a step further by providing a formal robustness guarantee for any input perturbations, for example, a classifier is said to be certifiably robust if for any input x, one can easily obtain a guarantee that the classifier's prediction is constant within some set around x [92].

Poisoning Defence:

Byzantine-resilient Defense. It is a popular defense mechanism against untargeted model update poisoning attacks. Various robust aggregation methods are provably effective defense mechanism under this category [27]. Using a robust aggregator to detect malicious client updates lowers an FL model's susceptibility to poisoning attacks.

Outlier Detection. On the other hand, explicitly identifying and denying malicious influence is a more proactive form of defense against poisoning attacks. Approaches include rejecting updates with too large error rates, measuring the distribution of parameter updates or looking for dormant neurons that are not frequently activated [93]. Outlier detection is a defense mechanism that increases the probability of finding malicious updates and hence lowering the possibility of poisoning attacks.

Empirical Robustness:

We can measure empirical robustness by implementing a model poisoning attack: a typical poisoning attack is to implant backdoor triggers to alter some local data (data poisoning) or the gradients (model poisoning) [91]. A mathematical explanation of how a model replacement attack work is provided by Wu et al. [93] and illustrated as follows:

It is assumed that least one compromised client could apply the backdoor patterns to administer a model replacement attack. Let w_t be the current model and N be the number of all clients. The global model w at time $t+1$ is an averaged mean of model updates from N clients at time $t+1$, and the goal is to replace the global model w at $t+1$ with the attacker's model x_{atk}:

$$x_{atk} = w_{t+1} = w_t + \frac{1}{N}\sum_{i=1}^{N}(x_{t+1}^i - w_t) \tag{3.1}$$

Now let x_{t+1}^m be the update from the malicious client m at time $t+1$, then by rearranging equation 3.1, we have:

$$x_{t+1}^m = N \cdot x_{atk} - N \cdot w_t - \sum_{i=1}^{N-1}(x_{t+1}^i - w_t) + w_t \tag{3.2}$$

Assuming $\sum_{i=1}^{N}(x_{t+1}^i - w_t) \approx 0$ [94], then we have the attacker's update be simplified as the following:

$$x_{t+1}^m = N \cdot (x_{atk} - w_t) + w_t \tag{3.3}$$

Certified Robustness:

The idea of the robustness guarantee is to define an "attack lower bound" which is the least amount of perturbation that is required for the attacker to succeed. Or in other words, a model is certifiably robust for an upper bounded amount of perturbations. Many certified robustness approaches and techniques have been proposed. Alfarra et al. [67] showed that certification technique like randomized smoothing [92] can be used to certify the robustness of a ResNet18 trained on FL setting. Weng et al. [95] developed the CLEVER metric (short for Cross Lipschitz Extreme Value for nEtwork Robustness) using the local Lipschitz constant for neural networks. CLEVER is an attack-agnostic derivation of the universal lower bound on the minimal distortion required for a successful attack. The CLEVER metric is available in the ART Python library mentioned in section 2.1.2.

3.1.2 System-level Robustness

System-level robustness should be considered in productionalized FL models where proper software development and deployment standards are expected and observed. Conventionally, testing is an essential approach to evaluate and enhance the robustness of any software system [22]. Fuzz testing [96] and bug bounty programs [97] are effective approaches to detect vulnerabilities of a system from malicious inputs. As FL systems involve more actors and more communication steps, there are more chances and attack surfaces for adversaries. Continuous testing and validation are key to ensure that the system behaves as intended. A robust system should also have fallback plans when things go wrong. For FL, there is a high risk of uncertainty and errors when training in a distributed environment. A reliable FL system should be able to mitigate the risks to ensure an efficient and effective collaborative learning process.

Functional Robustness:

Testing Coverage. There are various testing methodologies for robust system delivery, ranging from code review, unit testing, integration testing, system testing and acceptance testing. The purpose of testing is to make sure that the developed functionaries adhere to the requirements. The reason why this is important to FL is that an FL model requires more procedures like broadcasting messages to distributed clients, client selection and model aggregation. During development, developers should make sure that the functions for client selection are selecting the correct clients based on the required criteria and the functions for aggregation are following the aggregation steps correctly. An important metric in testing is the coverage, which is the percentage of code, branches or features that have been fully tested. Conventionally, the higher the testing coverage, the more verified the system is in terms of functional robustness.

System Reliability:

Error Rate. The key elements of reliability include the probability and the duration of time of failure-free operation [98]. A simple evaluation approach is to calculate the failure rate with is the number of failures over a given amount of time.

Maximum Timeout. The ability to recover from failure with error handling mechanisms is also an important reliability feature. In an FL configuration, there is the practice to define a maximum timeout duration and a dropout mechanism that determines how long the server should wait to receive model updates from the clients.

Dropout Rate. With the maximum timeout defined, straggler client nodes are often dropped to speed up convergence and optimize resources. In a way, a high dropout rate can also indicate a less reliable FL system that needs to improve on either network stability or the client selection effectiveness.

3.1.3 Algorithm-level Robustness

At the algorithm level, robustness evaluation is mostly focused on performance and generalizability. Performance benchmarking is a widely used approach to showcase how good an ML/DL model is. However, good performance does not necessarily imply generalizability which is a more long-term robustness indicator. In fact, generalizability is a major challenge in FL because each client has different local data heterogeneity and the aggregated global model might not be able to capture the data pattern of each of them. Non-IID data especially class imbalance may cause severe learning divergence to parametric models mainly in HFL [99]. Therefore, having a mechanism to deal with heterogeneous data is a desirable and recommended design pattern in FL [52]. There are data-based and algorithm-based approaches to tackle Non-IID data. In FL research, the algorithm-based approach focuses on personalization.

Performance:

Average Test Accuracy. The performance of an ML/DL algorithm is measured by test/validation loss and accuracy. In FL there are two approaches of measuring the performance. One is by reserving a set of test or validation data on the server side to be used for global model evaluation. The other is by evaluating test accuracy at each client's device (using client test data and the global model) and aggregating the test accuraries on the server side.

Personalization:

Several methods have been proposed for this category.

Regularization. Regularization is one of the personalized FL approaches which aims at minimizing the disparity between the global and local models.

Multi-tasking Learning. This is a method where multiple learning tasks are solved at the same time. This is beneficial for FL models by the nature of collaborative learning. Multiple organizations participating in the FL models can be training their personalized models to achieve better performance. For example, MOCHA [100] generates separated but related models on local client devices using data of related tasks.

Clustering. Clustered Federated Learning (CFL) groups client populations into clusters based on their similarity [101].

Personalized Layer. Another personalization method for neural network models is introducing personalized layers for each client in the model. FedPer [102] is such an example that uses the base layers as the shallow layers and personalized layers as the deep layers while keeping everything else the same as the baseline FedAvg algorithm. The experiment results showed that FedPer can achieve higher test accuracy than FedAvg.

3.1.4 Client and Data Reliability

This additional notion of robustness deals with client and data reliability in terms of client reputation and data quality. In the study by Kang et al. [103], the concept of reputation metric was introduced to gauge the trustworthiness of client workers. The reputation metric is supposed to help detect malicious updates early and to ensure that selected clients can be trusted. High quality data are essential for the development of machine learning models and even more so for FL models with inconsistent data qualities among clients. In the work by Pejó and Biczók [104], a novel approach was proposed to infer the data quality in an FL environment taking advantage of the round-wise training mechanism.

Scale:

Number of Clients. The scale of project also impacts the reliability of the system. In the case of FL, the number of clients determine the number of distributed network, devices or machines that need to be working together seamlessly to co-train an ML/DL model. In different FL settings, the scale is different. For example, a cross-silo VFL between a few organizations could be considered a smaller scale (10 to 100 clients) than a cross-device HFL setting among millions of mobile phones. With more clients, the network stability, computation power and availability of clients are all influential factors to the success and reliability of the FL system. Therefore, scale in terms of the number of participating clients is used as an approximation of how reliable the FL model is.

Client Reputation:

Reputation Score. In the work by [103], a novel reputation metric was proposed and a subjective logic model was used to calculate reputation score for each client interaction. After each training iteration, the server uses a poisoning attack detection scheme and the elapsed time to determine if the local update from the client is reliable. Reliable updates are treated as positive interactions and improves the reputation value, and vice versa. When the reputation value is below a certain threshold, then the client is treated as malicious and unreliable.

Data Quality:

Inferred Quality Score. Similarly, in the proposed method [104], after each round of training, the local updates are compared with the current global model to see if the new local updates are better or worse (or the same as) the global model. The inference is that if a round of training at one client improves the performance of the global model, then the data from that client has a higher quality score, and vice versa.

3.1.5 Limitations

For building resilience to attacks, existing defenses against backdoor attacks requires access to or analysis of training data and local updates. Evaluating the empirical robustness by administering attacks to the system also requires access to the training dataset to insert perturbations. The data access requirements may not hold in an FL setting where data privacy should be protected.

For system-level robustness, testing coverage alone cannot guarantee the degree of functional verification because the quality of the tests are not considered. It is difficult to use a generalized definition for failure rate as failures are defined differently according to different system requirements and some failure reports may not be triggered in the system. Emphasis on system reliability in terms of efficiency may also sacrifice fairness when certain clients with slower network speed or computation power do not get selected or get dropped more frequently.

For algorithm-level robustness, evaluating global model performance on the server side with even a small amount of reserved test/validation data may reveal sensitive client information to the server. Participating clients need to reach an agreement on test data usage for this type of global performance evaluation. Furthermore, aggregating the mean test accuracy of the global model from clients may not be accurate enough for evaluating the performance of personalized FL models [105].

For client reliability, calculating the reputation value at each round of training adds a large amount of overhead in a large scale FL system at the expense of resource cost. Similarly, comparing each local update with the global update at each round of training for measuring data quality is also a costly task.

3.2 Privacy Pillar

The biggest driving force for the development of FL has been data privacy because the whole point is to train ML/DL models over distributed data that could be sensitive and should be kept confidential. It is important that an FL model keeps up the promise of preserving data privacy within its lifecycle in order to gain trust from the training participants and users. Even though FL already elicits a degree of data privacy by definition, assumptions have to be made about the integrity of the multiple actors and parties in the system. If clients are honest but the server is honest-but-curious, prevention of information leakage from model parameter exchanges between clients and the server needs to be in place. If clients are honest-but-curious in peer-to-peer or VFL settings, then prevention of information leakage should focus on secure communication between clients. Moreover, information can still be leaked by malicious attacks from the outside.

Privacy-preserving techniques is a big topic in Trustworthy AI and FL. The comprehensive survey by Yin et al. [106] summarized the state-of-the-art PPFL approaches which served as a reference for this work for the evaluation of privacy-preserving capabilities of FL models. The representative techniques from the three major categories are used as a checklist to verify if an FL model has certain privacy defense.

Furthermore, the effectiveness of the privacy-preserving mechanisms need to be assessed to provide a full picture of the degree of privacy preserved. The list of technical privacy metrics provided by Wagner and Eckhoff [38] is helpful for evaluating privacy in ML/DL models.

3.2.1 Privacy-preserving Approaches

Thanks to the active research on privacy in AI, various privacy-preserving techniques have been proposed and some of them are shown effective in preventing information leakage or protecting data privacy in FL setting. Employing these techniques in an FL system can show that the system is committed to protecting privacy and has the technical capability to do so.

Perturbation-based:

Differential Privacy. The key idea of perturbation-based techniques is to add noise to the raw data so that the perturbed data are statically indistinguishable from the raw data. The most widely adopted scheme in this approach is differential privacy (DP). In FL there are global and local DP. Global DP requires a trusted central server but is more accurate than local DP. Local DP adds noise to the clients' gradient updates to conceal the true values from attacks during model exchanges or aggregation. Employing global or local DP in an FL system will preserve the data or gradient information to some extent depending on the amount of noise added.

Encryption-based:

Homomorphic Encryption. The most widely adopted scheme in this approach is homomorphic encryption (HE). Using additive HE on gradients will further prevent information leakage during model exchanges. Another popular scheme is SMC that enables distributed participants to collaboratively calculate an objective function without revealing their data. SMC is particularly useful in VFL setting where parties have to share intermediary gradient results.

Anonymization-based:

K-anonymity. The most widely adopted schemes in this approach are k-anonymity and l-diversity. K-anonymity is satisfied if each sample in the dataset cannot be re-identified from the revealed data of at least $k-1$ samples. L-diversity extends on k-anonymity so that the sensitive attributes of the samples are protected. Choudhury et al. [73] showed that the k-anonymity scheme could effectively preserve privacy even better than DP in an FL model.

3.2.2 Information Gain/Loss

The most direct evaluation of privacy protection effectiveness is the amount of information gain or loss which is quantified by the amount of privacy lost by the users or the amount of information gained by the adversary due to leakage or disclosure of information [38]. The state-of-the-art privacy metrics, to name a few, include Amount of Leaked Information (counting the information items disclosed by a system), Relative Entropy (measuring the distance between two probability distributions), Mutual Information (quantifying the shared information between two random variables) and so on. Liu et al. [74] developed a quantitative metric based on mutual information to evaluate privacy leakage in FL which is used in this work.

Information Leakage Risk:

In an FL system, model gradients are shared either between clients and server (HFL) or among participants (VFL) and the gradients can carry enough information for adversaries to reconstruct the model or infer original data. Liu et al. [74] developed a hierarchical mutual information estimation method, H-MINE, to measure the mutual information between the high-dimensional gradients and batched input data. The proposed metric can reflect the extent of information leakage during training when clients send the updated local gradients to the server for aggregation. The H-Mine algorithm can be seen in appendix A.6.

3.2.3 Uncertainty

Even with information leakage, the uncertainty of estimation by the adversary can still make a difference to the level and effectiveness of privacy breach. Under normal circumstances, high uncertainty in the adversary's estimation correlates with high privacy. Many uncertainty metrics are based on entropy which is used in this work for this category as well.

Entropy:

In general, entropy measures the uncertainty in predicting the value of a random variable. In FL, an adversary may be interested in identifying which data samples belong to a particular client or organization participating in the training. The entropy of X where X represents a participating client can be expressed as follows:

$$priv_{ENT} \equiv H(X) = -\sum_{x \in X} p(x) \log_2 p(x) \tag{3.4}$$

where $p(x_i)$ is the estimated probability of this client being the target.

3.2.4 Indistinguishability

In other cases, the adversary may be interested in distinguishing between two data samples of interest, and indistinguishability metrics analyze whether an adversary is able to do to. Privacy is high if the adversary cannot distinguish between two outcomes of the model. DP is a popular mechanism to enable indistinguishability in the training data by adding random noise. It is a formal statistical guarantee that any disclosure is equally likely whether a sample is in the dataset or not. The formal DP proof for privacy mechanism, a randomized function K, is to check if the output random variables for two datasets D_1, D_2 that differ at most to some extent (e.g. one row of data) differ by at most $exp(\epsilon)$:

$$priv_{DP} \equiv \forall S \subseteq Range(K) : p(K(D_1) \in S) \leq \exp(\epsilon) \cdot p(K(D_2 \in S) \tag{3.5}$$

Since the proof requires knowing if the model outputs came from two different datasets, it is difficult to evaluate the effectiveness of global or local DP in FL without any knowledge of the training dataset.

Global Privacy Risk:

Mehner et al. [39] devised a worst-case scenario analysis method to evaluate a global privacy risk by reducing the complexity of a DP model. They generalized the interpretation of privacy risk based on ϵ and the number of participants only. Based on their work, the simplified global privacy risk for a DP-based privacy mechanism is given by:

$$P = \frac{1}{1 + e^{-\epsilon}} \tag{3.6}$$

where P indicates the success probability of identifying the target.

The selection of ϵ in private machine learning with DP is often a difficult task. As seen in the worst-case global privacy risk, the value of ϵ determines how much noise is added to the raw data and how much protection the DP policy has. In FL, there is also the consideration of global versus local DP. Global DP is applied at the server side on the aggregated model parameters whereas local DP is applied at the client level with local model updates. The study by Wei et al. [107] proposed the Noising before model Aggregation FL (NbAFL) approach to add artificial noise to the parameters at the client's side before aggregation happens. They performed a few comparative experiments and showed that the lower range of ϵ which has higher privacy protection is below 10 and the higher range from 50 to 100 had poorer privacy protection. The NbAFL techniques is used in the experiments of this work and so are the ϵ ranges for metric evaluation.

3.2.5 Limitations

Regarding perturbation-based privacy-preserving techniques, a global DP can be effective if the central server can be trusted. However, if the server or the third-party cannot be trusted, then local DP needs to be added on the client side, resulting in a much larger totla amount of noise. Although perturbation-based techniques are simple and effective, they may degrade the data utility and decrease model accuracy. It is also known that the decrease in accuracy caused by adding noise is distributed unevenly across classes which explicably decreases fairness. Regarding encryption-based techniques, the disadvantages of homomorphic encryption and SMC are computation overhead and high communication costs. Anonymization-based techniques are vulnerable to inference attacks on the sensitive attributes.

For information gain/loss, the study [74] suggests that there are some inherent factors affecting the information leakage risk. One is that at the beginning of the training, the information leakage is usually high because the model has not yet been fitted to the data. The leakage reduces over time during the training as the model stabilizes. The other factor is that the inherent data distribution could also affect the risk of information leakage. The more unbalanced the dataset is, the more information could be leaked from the gradients. Furthermore, the proposed H-MINE framework runs two multi-layer perception models during the training to calculate the information leakage risk, which could be costly too.

For uncertainty, although entropy is an intuitive interpretation, it is not always an accurate representation of uncertainty because it is easy to construct different probability distributions that yield the same entropy value. For example, a uniform distribution over 20 clients and an almost uniform distribution over 101 clients where one has a probability of 1/2 has the same entropy [38]. Therefore it is difficult to use entropy to compare the uncertainty between different systems.

For indistinguishability, the generalization of DP by using the worst-case scenario, though more understandable and privacy-preserving, may be highly pessimistic as it estimates the maximal privacy risks for varying ϵ values independent of other parameters. Other metrics with more accurate and realistic estimations should be explored in this category.

3.3 Fairness Pillar

Putting trust in an AI-powered system is no different than putting trust in the underlying systems like companies, government, financial institutions or law enforcement where one believe should do the right thing. An important facet of doing the ring thing in human society is making just decisions with non-discrimination and fairness. In fact, a lack of fairness had been demonstrated in AI application and research when credit loan applications were found to favor certain demographic groups or facial recognition applications reflected racial discrimination. In the EU guideline for Trustworthy AI [20], fairness is listed as one of the four ethical principles rooted in fundamental rights that an AI system must respect in a trustworthy manner.

In fair AI [19], fairness is broken down into group-level fairness and individual-level fairness, each with various definitions. Group-level fairness means members of a particular group should not be subject to discrimination or demographic bias, which is often measured by the difference in classification accuracy among different groups. Individual-level fairness, on the other hand, means that similar individuals should receive similar treatment independent of their group membership. Whether an ML/DL model is trained in a classical centralized way or in a federated setting should not change the requirement that the model should strive to maintain both the group-level and the individual-level fairness.

One main source of unfairness in AI models are coming from the data. Different clients may be contributing different amount of data with various quality levels and class distributions. When data are not representative of the wider population then there is a selection bias which can then be propagated into the model. The selection bias can be manifested by feature distribution skew or label distribution skew [27], both of which are major challenges in FL where heterogeneity and randomness are multiplied by the large number of clients in the learning process.

Another perspective of fairness special to FL is the contribution fairness. Since multiple clients are training a global model together, the payoff or reward for each client should be proportional to their data contribution. Fairness in this respect is important for gaining trust from the clients and encouraging them to actively contribute to the learning process. Some approaches leveraging on concepts from game theory were proposed to devise fair incentive and reward mechanisms in FL.

3.3.1 Client Selection Fairness

An FL system can be of different scales. In the cross-silo VFL setting, there are typically $2 - 100$ clients representing different organizations. In a cross-device setting, the scale could be as massive as up to 10^{10} clients representing edge devices. Usually in FL, not all the clients would participate in every round of training due to various reasons such as availability, communication costs, etc. Only a fraction of clients are selected to participate in training process in each round. In the basic case, the client selection scheme is to randomly sample the specified fraction of clients. In practice, there are more criteria than random selection, such as the availability, network speed, computation power, battery level of the clients' machines. In the worst case, clients located in regions with low network speed or models with weaker computation power could never get selected and represented in the training data. Under representation is a source of selection bias that could lead to unfair model outcome towards the under represented group.

Participation Rate Variation:

In statistics, Coefficient of Variation (CV) is the measurement of how far the data values from a set are dispersed from the mean. It is used to analyze the distribution of participation rates among all clients. The assumption here is here is that, with similar clients, the more dispersed the distribution of participation rate, the less fair the client selection mechanism is, and vice versa.

3.3.2 Performance Fairness

Even though performance fairness already exists in AI as in individual and group fairness, in FL, another grouping needs to be considered which is the client-level fairness. Li et al. [108] suggest a definition that a model provides a more fair solution to the FL learning objective on the clients if the performance is more uniform than that of another model.

Accuracy Variation:

The test accuracy is used as a representation of performance. The aggregated global model and test data from each client are used to measure the test accuracies. The more uniform the test accuracies among the clients, the more fair is the model performance.

3.3.3 Group-level Fairness

Evaluating and mitigating demographic bias in FL is more difficult than in centralized learning, First of all, raw training data, labels and sensitive demographic information of each participant cannot be revealed. Second, in centralized learning, all the training data can be analyzed and pre-processed to balance the class distribution before training, whereas in FL, different clients pre-process their data locally so additional mechanisms are needed to adjust the global data distribution in a secure and protected manner.

Discrimination Index:

In the work by Zhang et al. [77], the discrimination index metric was used. This metric measures the difference in the F1 score between a particular demographic group and the rest in the population. The metric value falls between $[-1, 1]$ where the ideal discrimination index should be as close to 0 as possible. To calculate this index globally would reveal sensitive attributes and statistics of the demographic group in the client data. The work proposed using secure aggregation meaning that the clients send their calculated discrimination indices to the server which then uses secure aggregation method to combine the indices together into a global value.

3.3.4 Class Distribution

As mentioned in section 3.3, the heterogeneity problem in class distribution can be closely related to client selection fairness. Class imbalance goes hand in hand with unfavorable client selection, resulting in a slow converge rate of the global model. Analyzing the class distribution of the training data used in an ML/DL model provides insight into whether the data samples are selected properly to reflect a fair representation of the wider group. In theory this should apply to FL models as well except that in practice this often requires access to and analysis of the raw training data, which goes against the priority of privacy protection in FL. Yang et al. [109] proposed an estimation scheme to reveal the class distribution without the awareness of raw data specifically for FL setting. Secure aggregation can also be considered for aggregating class distribution information among clients.

Class Imbalance:

Two approaches can be used to evaluate the class imbalance. One is the estimation scheme using an auxiliary well-balanced dataset and the gradients of a DNN model [109]. When the auxiliary data samples are fed to the updated model, we can obtain gradients vector like so:

$$\{\nabla L^{aux}(w_1), \nabla L^{aux}(w_1), ..., \nabla L^{aux}(w_C)\} \tag{3.7}$$

where each $\nabla L^{aux}(w_i)$ is related to the ith neuron and class C_i. When training DNN in classification tasks, the gradient square for different classes have the following expected approximation relation:

$$\frac{E \parallel \nabla L^{aux}(w_i) \parallel^2}{E \parallel \nabla L^{aux}(w_j) \parallel^2} \approx \frac{n_i^2}{n_j^2} \tag{3.8}$$

where L denotes the cost function of the neural network and n_i and n_j are the number of samples for class i and class j. This theorem reveals the correlation between the gradients and class distribution. Then for class C_i, the estimation of class ratio can be defined as:

$$R_i = \frac{\frac{\beta}{e^{\|\nabla L^{aux}(w_i)\|^2}}}{\sum_j e^{\|\nabla L^{aux}(w_i)\|^2}} \tag{3.9}$$

where β is a hyperparameter that can be tuned to control the normalization between classes. Once we obtain the composition vector $R = [R_1, ..., R_C]$ that indicates the distribution of raw data, we can use the Kullback-Leibler (KL) divergence to evaluate the class imbalance of each client like so:

$$D_{KL}(R \parallel U) = \sum_{i \in C} R_i \log \frac{R_i}{U_i} \tag{3.10}$$

where U is a vector of ones with magnitude C. When two distributions perfectly match, the KL divergence equals 0, otherwise it can take values between 0 and ∞. Therefore, the closer the KL value is to 0, the more balanced the class distribution is. This estimation scheme can only apply to neural networks.

Another more general way to get the class imbalance information in FL is to ask every client to submit their class distribution to the server where secure aggregation method combines all the class distribution into one unified distribution. Then the coefficient of variation of the class distribution can be used to calculate the level of variation of the class sample sizes to determine the class imbalance level. This more generalized approach is used in this work to evaluate class imbalance level in an FL model.

3.3.5 Limitations

For client selection fairness, enforcing equal participation rate without considering the other qualities of the clients may result in unsuitable or ineligible clients being selected. The Architectural Soundness pillar in the later section 3.6.1 would explain that a client selector scheme with predefined criteria can be used to make sure that the selected clients fulfill certain requirements for the particular learning task in order to optimize resource usage and performance. Therefore, there is an inevitable trade-off between client selection fairness and suitability.

For performance fairness, the measurement of accuracy variation does not fully explain the fairness in personalized FL models. It is suggested to study the per-client personalized accuracy and the accuracy improvements among clients with an equitable notion of fairness [105]. New fairness metrics need to be explored for personalized models.

For group-level fairness, although the global discrimination index or a particular demographic group can be calculated through secure aggregation, how to coordinate clients in calculating their local discrimination index of the demographic group without revealing the sensitive attribute of the group remains unclear.

For class distribution, the proposed privacy-preserving estimation scheme only works for DNN models where the gradient square to class ratio approximation holds. Aggregating the class distribution among clients may still leak some degree of private data information in terms of the sample size per class. Furthermore, the unified class distribution among all clients may not reflect the individual class imbalance within each client's dataset where personalzation is concerned.

3.4 Explainability Pillar

The AI HLEG of the European Commission deem explicability as a crucial building block for user's trust in AI system. Their trustworthy AI guideline demands that AI processes should be transparent, with the capabilities and the purpose of AI systems openly communicated and decisions explainable to those directly and indirectly impacted. The reason is that without such transparency, an AI outcome or decision cannot be fully justified as lawful, ethical and respectful to human rights.

In fact, according to the guideline, transparency is one of the seven key requirements for the realisation of Trustworthy AI. Transparency is often expressed as interpretability which is often wrongly mistaken as interchangeable with explainability. Interpretability is the passive characteristic of a model referring to the level of understandability for humans, whereas explainability is the active characteristic of a model referring to any action or procedures taken by the model to clarify the intent of its internal functions [42]. Needless to say that both interpretability and explainability are important facets of transparency. The early AI systems were fairly interpretable, but with the rise of black box algorithms that are not easily interpretable for human understanding, explainability became highly demanded from various stakeholders in order to seek confidence and trust in the system because humans are generally reluctant to adopt technologies that are not directly interpretable.

Overall, explainability is the ability to provide an explanation about the technical process of an AI system and the resultant decisions in the applications, and interpretability facilitates part of the explanations from the perspective of model design. Interpretable models can be explained by analyzing the model itself but for non-interpetable models, post-hoc explainability methods are diverse means to enhance the interpretability.

For FL, since ML/DL models are also used in the training process, the requirement of explainability for the algorithmic model also applies. However, as mentioned in section 2.3.1, the privacy constraint makes it difficult to directly access and analyze any raw data or protected model features in some FL setting. Researchers had to come up with a privacy-preserving version of SHAP for explaining VFL models [80]. Since FL is also larger in scale with additional processes, like the distribution of tasks to decentralized local models, collection and aggregation of model updates, privacy mechanisms and so on, more explanations have to be made about these processes in the pipeline regarding how they affect the output of the system.

3.4.1 Interpretability

As mentioned before, some ML/DL models are interpretable by design and some are not, depending on the algorithm. The comprehensive review on XAI [42] categorizes ML/DL models by their transparency level which can be used to indicate the interpretability of the models. In literature, model size is also widely used as a measure of interpretability [110]. The definition of model size could vary depending on the algorithm design.

Algorithmic Transparency:

A model is considered transparent if it is understandable by itself. This definition could be very subjective to different levels of intellectual grasp. However, algorithmically transparent models must first be fully explorable by means of mathematical analysis and methods, and then the assessment considers model complexity (in terms of the number of variables and interactions) and decomposability (in terms of interpretability of each component of the model) [42]. In summary, the recognized interpretable models are: linear regression, logistic regression, decision trees, decision rules, k-nearest neighbors (KNN), Bayesian models. Even within the interpretable models, the level of interpretabiilty slightly varies as seen in the table (appendix A.1). The non-interpretable models include tree ensembles, support vector machines (SVM), multi-layer neural network (MNN), convolutional neural network (CNN) and recurrent neural network (RNN).

Model Size:

Different algorithms have different definitions of model size. For instance, it could be the number of decision rules, the depth of a decision tree, the number of features in a linear/logistic regression model, the number of trainable parameters in a neural network, etc [110]. The larger the model size, for example, the higher the number of features and interactions, the harder it is to understand and explain the causal relationship between input and output. This is because with more features and interactions there is a more combined influence on the prediction outcome.

3.4.2 Post-hoc Explainability Methods

The three most common post-hoc explanation methods are simplification, feature relevance and visualization [42]. Simplification means reducing the complexity of a non-interpretable model into a simpler surrogate model that is more easily interpreted and understood. For example, in the study by Haffar et al. [111], they added explainability to an FL black box model by using random forests containing decision trees to compute the importance of the features in predictions. Given that the simplification of multi-layer neural networks becomes more complex as the number of layers increases, explanation by feature relevance becomes more popular. Feature relevance explanation method measures the influence, relevance or importance of each feature in the prediction output, hoping to explain what aspect of the training population cause certain behaviour of the model. The most recognized technique is SHAP [45] that calculates an additive feature importance score for each particular prediction. Visualization method provides visual representations of AI models to facilitate better understanding by humans. For example, a linear regression model can be represented by a relationship diagram between the dependent and independent variables. This type of visualization is more achievable with interpretable Machine Learning (ML) models than with high-dimensional DL models. For black box DL models, novel visualization methods have been proposed using sensitivity analysis [112]. Additionally, the mean square error per epoch diagram and the receiver operating characteristic (ROC) curve are common performance evaluation plots in ML.

Feature Importance:

It is argued in the work by Wang [80] most of the model explanation methods can be directly used for HFL because all participants share the full feature space in their local data. However, even for HFL, exposing feature information to the server for calculating feature importance score is not ideal given that the server may not be honest. For VFL, methods like SHAP cannot be directly used because parties do not share the full feature space. The idea of the variant version of SHAP for VFL [80] is to combine one party's features into a federated feature space when being referenced by another party for the feature importance calculation. The algorithm can be seen in appendix A.3. The Shapley values of features can be either positive or negative. A large positive values indicates that the features has a high positive impact on the outcome prediction whereas a negative value indicates negative impact from the feature. By using the absolute Shapley values of features, we can get a list of feature importance strengths and calculating the variation among theses strengths can give us an idea how spread out the feature impacts are. The key idea for this metric is that the more spread out the feature importance scores are, the higher the chance that we have some feature with very high or low explanatory power which indicates that the outcomes can be explained by the big difference. On the contrary, if the feature importance scores are uniform, which implies that no feature stands out, then the level of explainability by feature is low.

Visualization:

In fact most visualization techniques for DL-based classification models resort to the visual representation of the feature importance by illustrating how each feature impacts certain prediction output [42]. Two examples of feature importance visualization can be seen in appendix A.5 and A.4. For explainable FL, the study by Ungersböck et al. [113] proposed a lifecycle dashboard that visualizes information from the server, starting from client registration to training, validation and deployment. The dashboard shows which clients participated at which round of training and the current status of the model. This type of dashboard representation approach follows the status of the FL system and paints a good picture of how the aggregated global model evolves throughout the lifecycle.

3.4.3 Limitations

For interpretability, measuring algorithmic transparency requires analyzing the model architecture and parameters which could be confidential in productionalized industry environment. There is the trade-off between preserving the intellectual property and the explainability of AI systems.

For post-hoc explainability methods, the popular feature relevance technique has not been shown effective in multi-layer neural networks. The challenge is defining the relationship between the network output and the input neurons in order to assign a correct feature importance score to the proper feature component.

3.5 Accountability Pillar

Accountability is one of the seven key requirements for the realisation of Trustworthy AI defined by the EU guidelines for trustworthy AI [20]. It is the state of being responsible and answerable for a system about its behaviours and potential impact [114]. The first step towards accountability is good documentation where claims about the AI system and procedures are logged and verified throughout the life cycle of the process. The documentation should contain evidences of the 5W: what, where, when, who, why. These evidences can help clarify how an ML/DL model is built and how it arrives at certain outcomes. Good documentation provides transparency and instills trust in the system.

IBM Research was the first to propose a FactSheet to record facts about the overall ML/DL pipeline [48]. Subsequently, they extended the FactSheet approach to enable accountability in FL as well [85]. The accountable FL FactSheet template [115] is a comprehensive document that contains meta information about the project, the participants, the data and the model configurations and performance. This thesis uses a simplified and extended version of the template as a way to quantify document completeness.

Another important aspect of accountability is monitoring because even with complete and detailed documentation, every responsible party has to make an effort to ensure that the system is built strictly following the intended architecture, development and deployment processes. In literature, auditing is a widely proposed approach to monitor accountability of an AI system.

3.5.1 FactSheet Completeness

As compared to traditional ML, it is even more challenging to achieve accountable FL because of the distributed nature of the system and the potential lack of trust among different parties involved in the training pipeline. In fact, the decentralized training methodology and privacy mechanisms make FL non-transparent by design. Since FL is more complicated in architecture and more privacy-preserving, the FactSheet should contain information about the additional layer of configurations and avoid sensitive information about participating clients. Based on the IBM Accountable FL FactSheet template, this work proposes a simplified but extended version with seven major sections that should be either extracted from the system or manually filled by administrators for the purpose of recording the basic facts about an FL system. The evaluation of the metrics under this notion uses a checklist based approach which means verifying if the content of each section can be found in the FactSheet.

Project Specifications:

This section of the FactSheet documents the overview, the purpose and the background of the project. The overview explains what the project is about. The purpose details the goals of the project and the background elaborates on the relevant information and knowledge that motivate the project.

Participants:

This section documents the information regarding the participants of the federated learning process. The original template suggests a table with participants' names and their organization unit names for identity verification. In practice, this should be taken care of by a dedicated client registry that manages all client information [52]. Therefore, in this version, the section contains more macro-level statistical information about the participants, for instance, the total number of clients, the sample client rate in percentage and the client selection scheme (name).

Data:

This section documents the information regarding the data used in the FL learning process. Two aspects are included: data provenance and pre-processing procedures. Providing data provenance can help trace the origin and the flow of the data to access validity and even reputation. Pre-processing steps can tell us how the raw data have been handled before being fed to the model for training. Depending on the FL setting, different clients may have different data pre-processing steps. For example, a bank and an e-commerce company collaboratively training a VFL model have different types of data that need to be handled differently. In that case, individual clients should report their data handling steps to the FactSheet.

Configuration:

This section contains all the important configurational information about the FL model. First of all, the type of optimization algorithm and the trainer ML/DL model should be mentioned. Then, the global hyper-parameters used by the aggregator should be defined, for instance, the number of rounds, the maximum timeout and the termination accuracy. Lastly, the local hyper-parameters used by the trainer at each client, such as the learning rate and the number of epochs, should be defined as well.

Performance:

This section showcases model evaluation results reported from the aggregator, for instance, the test loss, accuracy, and feature importance variance. The performance of the model is important information that can validate the utility of the model. Conventionally, a higher model performance correlates to a higher confidence and trust in the model.

Fairness:

This section documents the level of fairness of the FL model in terms of the variance of performance, client selection rate and class imbalance among the clients. These three aspects form a basic indication of how fair the model is among the participants.

System:

This section documents the system information for the learning process. It includes the average time spend on training, the model size, the model upload and download speed in bytes. This information gives an indication of the amount of resources expected to be utilized for the process.

3.5.2 Monitoring

Traditionally, monitoring of a system comes in the form of auditing. There are external and internal audits. External auditing is a process during which justification of the system is reviewed and assessed from the outside to ensure compliance with company policies, industry standards and government regulations. Internal auditing is usually more technical and related to quality assurance. For AI systems, there is algorithmic auditing which is a range of approaches to audit algorithmic processing systems. The evaluation of the metrics under this notion uses a checklist based approach which means verifying if the FL system employs any external or internal algorithmic auditing.

Algorithmic Auditing:

In the simplest form, algorithmic auditing can be testing like functional testing, performance testing, user acceptance testing, etc. It could also be system anomaly or attack monitoring. Some organizations even invite or hire external hackers to find vulnerabilities as a measure of monitoring. For a more systematic approach, the SMACTR framework proposed by Raji et al. [47] is a good option. It is an internal auditing framework that includes five stages of scoping, mapping, artifact collection, testing and reflection. Each stage yields a set of documents which together form an overall audit report. The process starts with the organization defining the values or principles that should be followed and upheld. Then along the pipeline, interviews, checklists, testing and retrospection are carried out considering the core values or principles defined in the scoping stage.

3.5.3 Limitations

The major challenges of the FactSheet approach are authenticity, quality of content and immutability. Many sections contain information that could be extracted from the FL model artifact. However, the method to automatically retrieve these information is difficult to be framework-agnostic as there is no guideline or standard for FL frameworks. For the manually entered information, we have to make sure the information is singed off by the appropriate person and role. Another limitation here is that the quality and the immutability of the content are not evaluated. A complete FactSheet does not automatically reflect accountability if the quality and the immutability of the content are not guaranteed. Blockchain technology has been widely proposed to enhance the accountability of FL, however, there is also a trade-off between transparency and privacy as not all information about the client data and model can be publicly documented. As for monitoring, there is still a lack of guideline on the proper framework of algorithmic auditing and quantifying the effectiveness of algorithmic auditing is difficult.

3.6 Architectural Soundness Pillar

The EU AI guideline [20] states that Trustworthy AI should provide the foundation for those affected by the AI system to trust that the design, development and use of AI are lawful, ethical and robust. Having a robust design that optimizes performance and ensures security instills trust in the system.

As compared to traditional centralized machine learning, an FL model has to consider both algorithmic design and architectural design because it is not just one ML/DL model but an entire distributed system that presents more architectural design challenges. The quality of algorithmic design have been covered and discussed from the perspectives of explainability and accountability. Architectural design focuses on how different components of the FL system should work according to best practices to produce a trustworthy FL model.

The major architectural design challenges of FL have to deal with communication, efficiency, resource limitation and security. To put in perspective, it is very challenging to coordinate the learning process among tens of thousands of client devices while ensuring model integrity and security. Global models might converge slowly due to heterogeneous client data. Inconsistent client network and limited resources might cause clients to drop out and training failures could impact model quality. Although there is active research in FL algorithms, there is still a lack of research and guideline on the architectural design of FL systems.

In the work by Lo et al. [52], a collection of design patterns suitable for the lifecycle of an FL pipeline was presented. Two key patterns each from the area of client management and model management were selected as metrics here. The evaluation method of the metrics in this pillar uses a checklist based approach which means verifying if the required design patterns are being used in the FL system.

3.6.1 Client Management

Just like how traditional machine learning needs to first collect or gather enough quality data, an FL model needs to first attract clients to participate in the collaborative training. Setting up incentive mechanisms to encourage clients' participation is topic for another discussion. Here we assume that we have enough interested parties willing to participate in the FL process. Then, a mechanism is needed to gather meta data about the clients so that the server can broadcast messages to the interested parties to start training. Managing client information requires a client registry where clients can register themselves for training. A client registry can verify the identities of clients and track client interactions with the server. Even with verified clients, it is not guaranteed that the clients' data are suitable for a particular learning task. A client selector can be used to filter eligible clients for training.

Client Registry:

The proposed design pattern maintains the client registry in the central server for the client-server architecture. The server sends a request for information along with the initial local model to the clients when they first connect to the system. The information requested includes device ID, connection up and down time, device computation power storage, etc. Having a client registry enables the system to manage client connections and track the status of all client devices.

Client Selector:

The proposed design pattern also maintains the client selector in the central server where the selection takes place. Before each round of training, the client selector actively selects a certain number of clients for the training according to predefined criteria for the purpose of reducing convergence time and optimizing the performance of the model. Having a client selector optimizes resource usage and reduces the risk of client dropout and communication latency.

3.6.2 Model Management

In a distributed learning process like FL, multiple rounds of training and aggregation of models generates numerous local model updates and aggregated global models during the process. For instance, running 100 rounds of training on 100 clients will generate 10,000 local models and 100 global models. Most of the time an FL experiment run will only record the evaluation results in each training round and produces the ultimate global model when training stops. Without recording the local models and the intermediary global models, there is neither traceability nor fallback when something goes wrong in the training process. A model co-versioning registry can help trace the model quality and improve system accountability. A common challenge in AI model management is dealing with concept drift which is the phenomenon when the performance of an ML/DL model degrades over time due to the unforeseen changes in the underlying data generation models [116]. This is a bigger challenge in FL where concept drift could happen to different clients' data samples at different times. A model replacement trigger is a mechanism that can detect model degradation and alarm the system to retrain the model.

Model Co-versioning:

A co-versioning system aligns the local model versions with their corresponding aggregated global models. It can be a registry where local model versions are stored and mapped to the associated global models. With this registry, model updates and aggregations do not always have to be synchronous because the server can refer to the mapping to perform asynchronous aggregations. Another advantage is that it provides the option of early stopping if a model converges before the specified number of rounds so that an intermediary global model can be used as the final global model.

Model Replacement Trigger:

A model replacement trigger works by detecting the global model performance dropping below a certain threshold level. It has to first compare the global modal performance in all clients to see if the performance degradation is a global event. If the degradation is global and persistent, then the a new global model training task is triggered.

3.6.3 Optimization

Algorithm:

As discovered in the literature review (2.2), FedAvg is considered the baseline aggregation algorithm. Several other optimization algorithms have been proposed as an extension to the baseline for various purposes. Based on the goal and the context of a FL model, the choice of optimization algorithm can impact the performance of the model. Various studies have conducted performance benchmarking of FL optimization algorithms. For example, Nilsson et al. [55] and Xie et al. [32] compared different FL algorithms to showcase the performance benchmarking of FL models. The comparisons serve as a reference for this metric.

3.6.4 Limitations

The major limitation of this pillar is how to quantify the quality of architectural design. Information on whether an FL system has certain design patterns is not readily available in the model artifact as it involves organizational and management decisions.

The client registry pattern may have a drawback on data privacy because recording the client device information on the central server can lead to privacy leakage. The client selector pattern may also lead to privacy issues due to requesting resource information from the clients. Excluding certain clients because of the selection scheme can also have fairness drawback which may lead to data heterogeneity issue and loss of model generality.

The model co-versioning strategy may incur expensive storage cost for storing the local and global models and their mapping. The drawbacks of the model replacement trigger are higher cost of computation for periodic performance evaluation and higher cost of communication for comparing the global model performance degradation among all clients.

3.7 Trustworthy FL Taxonomy

The six pillars of trustworthiness together constitute a comprehensive taxonomy describing the requirements for a trustworthy FL model. Figure 3.1 presents a visual representation of the taxonomy that combines the six pillars of trust explained in the previous sections - **robustness, privacy, fairness, explainability, accountability, architectural soundness**.

Under each pillar, the major aspects defining the pillar are grouped into notions, for instance, the robustness pillar has four notions - *resilience to attacks, system-level robustness, algorithm-level robustness, client and data reliability.*

Under each notion, the specific metrics that can be calculated to quantify the level of utility towards trustworthiness are defined.

This taxonomy serves as a basis for the evaluation and assessment of the trustworthiness level of an FL model. Customization such as adding, removing or modifying metrics based on individual system context is advised.

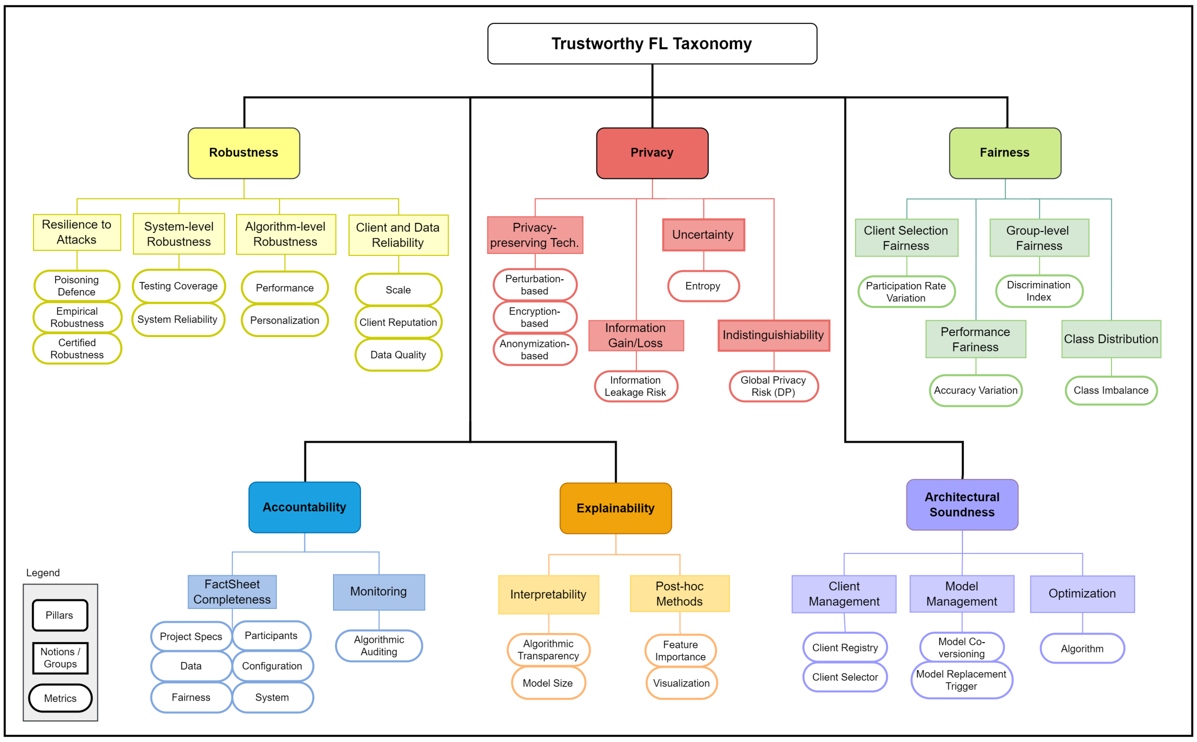


Figure 3.1: Trustworthy FL Taxonomy

Chapter 4

Trustworthy FL Evaluation Algorithm

The second contribution of this work is the design and implementation of a prototype evaluation algorithm, *FederatedTrust*, to quantify the trustworthiness level of FL models. The Trustworthy FL taxonomy presented in section 3.7 in Chapter 3 was used as the basis of the algorithm design. The goal is to create a light-weight, modular and extensible trust evaluation framework. So far this work is the first attempt to evaluate the overall trustworthiness level of an AI model in the federated learning context.

Extensive research and analysis was conducted on existing FL development and benchmark frameworks. Several design options were considered: 1) implement a new trustworthy FL framework with a built-in evaluation module, 2) extend on an existing FL benchmarking framework to incorporate additional trust metrics, and 3) implement a configurable trust evaluation module/library/package that can be deployed or imported by an existing FL development framework. The scope of option 1 is too large considering the focus of this work is on the evaluation part. Option 2 presents several challenges in extending existing benchmarking frameworks due to flexibility and requirement constraints. The final design decision was to go in the direction of option 3.

In fact, the initial goal was to build a framework-agnostic evaluation algorithm, however, it was found challenging given the requirements of this prototype because the evaluation of several pillars, mainly accountability and architectural soundness, requires extracting meta data about the FL model such as configuration details. Different FL frameworks have been developed differently meaning that there is no standard way to gather meta data among the frameworks. For demonstration purpose, one FL framework was selected as a reference framework for the design and implementation of the trustworthiness evaluation algorithm in this work. The long-term goal is that with extensibility and future work, the evaluation features of the algorithm can be extended, updated and consolidated for the use in multiple FL frameworks as well.

In the following sections, the design details and the structure of the evaluation algorithm are presented (4.1), followed by the explanations of the implementation with respect to the chosen FL framework and the selected metrics from the taxonomy (4.2). After that, the deployment of the *FederatedTrust* is demonstrated (4.3) with a chosen use case in the FL framework. Lastly, limitations of the algorithm are discussed (4.4).

4.1 Algorithm Design

Before designing the trustworthiness evaluation framework, it is important to understand the current state-of-the-art FL frameworks and their architectures. This work focuses on open-source FL libraries and frameworks that have been developed in recent years and are popular in the research community. The early FL simulations using popular ML libraries like TensorFlow and PyTorch usually synthetically generate federated clients by dividing a large dataset into a specified number of chunks and each chunk simulates a set of client data [28]. Later on, more realistic frameworks were developed using different processes [29] or docker containers [89] as representation of different clients. Event-driven architecture was also employed in a few frameworks [29][32] to properly simulate a distributed learning setting.

The architectures of FL frameworks determine the location and the kind of information available for evaluation. The availability and accessibility of input data from the framework for metric calculation is an important consideration for the design of the evaluation algorithm. For example, some frameworks do not expose the client selection process so that participation information can only be retrieved from the client side for evaluating fairness. Additionally, due to privacy constraint, the design has to define where each metric calculation takes place. To make such design decisions requires the analysis of how the FL framework performs messaging between clients and server and what are the practical assumptions according to industry standards. For example, does the server in HFL hold all the labels? Is there a global FactSheet and numerous local FactSheets? Where are the FactSheets stored? Questions like theses are essential for the design of the trustworthiness evaluation algorithm.

4.1.1 Exploring Open-Source FL Frameworks

Since the term Federated Learning was introduce in 2016, many FL development libraries, frameworks and platforms have been developed. The following list of FL libraries and frameworks was assessed:

TensorFlow Federated (TFF) [28]: It is a federated learning module introduced in 2018 by TensorFlow, a popular open-source library for ML. Although TFF has a well-maintained documentation and a suite of tutorials on their website, the module code has become incompatible with the latest version of Python due to a deprecated internal library.

PySyft [117]: It is an open-source multi-language library enabling secure and private machine learning by wrapping and extending popular DL frameworks such as PyTorch. It has been used to set up FL workflows and use cases since 2019 [118].

Flower [29]: It is a user-friendly FL framework developed by Adapt since 2020. A basic FL set up using Flower only takes less than 20 lines of code. Even though very easy to use, the simplicity also hinders the extent of customization and flexibility.

FLUTE [30]: It is an open-source industrial grade FL platform developed by Microsoft since 2019. It provides integration with Azure ML and can support high-performance large scale FL.

LEAF [31]: It is a basic performance benchmarking tool for FL developed in 2019. LEAF includes a suite of open-source federated datasets. The evaluation mainly focuses on performance and computation resource usage and the FL simulations are very basic without privacy or adversarial defenses.

FedEval [89]: It is a general-purpose benchmarking framework for FL developed in 2020. The framework has five evaluation metrics - accuracy, communication, time efficiency, privacy and robustness. Although FedEval has two overlapping pillars as this work, it is unclear how to add or customize evaluation metrics from the source code.

FedML [119]: It is a Federated Learning/Analysis and Edge AI platform developed as a start up in 2020. FedML aims to create a platform for the AI community can collaboratively build FL models on an open cloud. This is a platform for the FL community rather than a framework.

FederatedScope [32]: It is an open-source FL framework recently developed in 2022 by the Data Analytics and Intelligence Lab (DAIL) of Alibaba DAMO Academy [33]. FederatedScope employs an event-driven architecture to provide users with great flexibility. It is a comprehensive framework that builds on many existing research work in FL such as [31] and provides a suite of functionalities for privacy and adversarial attacks.

After careful comparison and analysis, FederatedScope was chosen as the reference FL framework for the following reasons:

1. Standalone mode and distributed mode allow for setting up clients in an experimental or realistic manner.
2. Data zoo contains a suite of federated dataset such as FEMNIST and Celeba from the LEAF study.
3. Algo zoo contains a list of state-of-the-art optimization algorithms such as FedAvg, FedProx, FedOpt, pFedMe, etc.
4. Model zoo contains a list of state-of-the-art computer vision and language models.
5. Configuration of differential privacy, personalizationa and privacy attacks are easy.
6. Steps to add customized evaluation metrics is straight forward and well-documented.
7. The framework was validated for its usefulness. A standalone experiment with 200 clients and 40% sampling rate trained a global CNN model on the FEMNIST dataset for 300 rounds and gave a test accuracy of 83.25%.

4.1.2 FederatedScope

Figure 4.1 shows an overview of the actors and steps within one round of FL training implemented in FederatedScope. Considering the basic cross-device client-server HFL case, there is one central server and N number of clients. The basic FL learning process is as follows:

1. Server selects a fraction of M clients and broadcasts the initial global model to each client
2. Once receive the global model, client $1, .., M$ perform local training using their trainer based on their private data
3. After local training, clients return the model updates to the server
4. With the help of an aggregator, the server performs federated aggregation on the received model updates, and optimizes the global model.
5. Steps 1 - 4 repeats for a specified number of rounds.

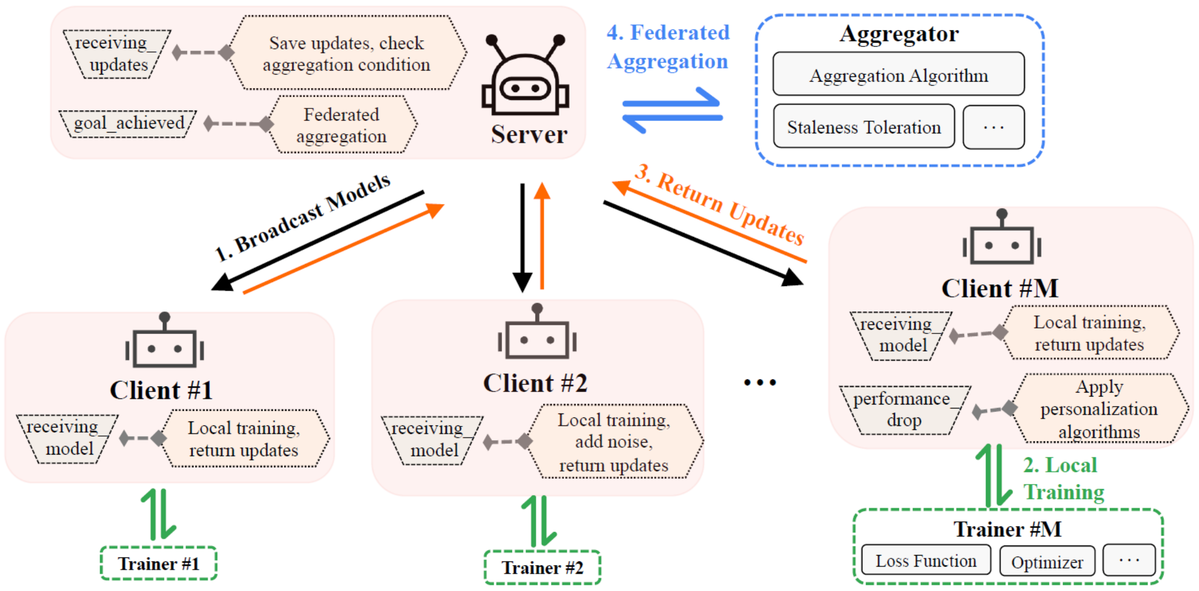


Figure 4.1: Overview of an FL round implemented with FederatedScope

4.1.3 Context and Assumptions

Based on the background knowledge of state-of-the-art FL and the architecture of the FederatedScope framework, the following contexts (C) and assumptions (A) are defined:

C-1: The basic use case that the algorithm should be developed for is a client-server HFL model.

A-1: The central server is honest and maintained by a trusted system owner.

4.1.4 Requirements and Constraints

Based on the goal of quantifying trustworthiness in FL, the basic functional requirements (FR), non-functional requirement (NFR) and privacy constrains (PC) are defined:

FR-1: Each of the six trustworthy FL pillars must be represented in the algorithm, meaning at least one metric from each pillar must be calculated towards the final score.

FR-2: The final trustworthiness score is a weighted average of the trustworthiness scores from all the pillars.

FR-3: The trustworthiness score of each pillar is a weighted average of the trustworthiness scores from all the notions.

FR-4: The trustworthiness score of each notion is a weighted average of the trustworthiness scores from all the metrics.

NFR-1: The algorithm should add minimal computation overhead and complexity to the FL model.

NFR-2: The algorithm should be modular and configurable.

PC-1: The algorithm is allowed to gather and process input data and export the processed data to a given output directory provided by the FL model.

PC-2: The algorithm is not allowed to store or download any data from the FL model within its local project.

PC-3: The algorithm is not allowed to share any data from the FL model with other entities.

PC-4: Any reports or documents generated by the algorithm do not leave the FL model framework.

PC-5: The calculations of metrics can only take place either at the clients' local devices or the central server not at any other third parties.

PC-6: If there is any aggregation of metrics from the clients at the central server, the aggregation should be performed securely if the individual client metrics contain sensitive information.

4.1.5 Architecture

Figure 4.2 shows an overview of the interactions between FederatedScope, the FL framework, and *FederatedTrust*, the trustworthiness evaluation algorithm. *FederatedTrust* is designed as a third party library that that be imported into the FederatedScope framework. This figure expands on the original overview diagram (Fig 4.1) by adding the missing steps and components that are part of the original framework.

The FederatedScope framework uses a "Config" file to store all the configuration parameters of the FL model. The framework also performs metric evaluation after every round of training be default. Once training is done, an "Eval Results" file is generated with the global model performance on all clients and the aggregated performance. These two files are important input documents for the *FederatedTrust* algorithm to calculate metric scores for a few pillars. Meanwhile, *FederatedTrust* generates a "Client Selection" file containing a map of encrypted client ids to their selection frequency. This file is generated duirng the training process.

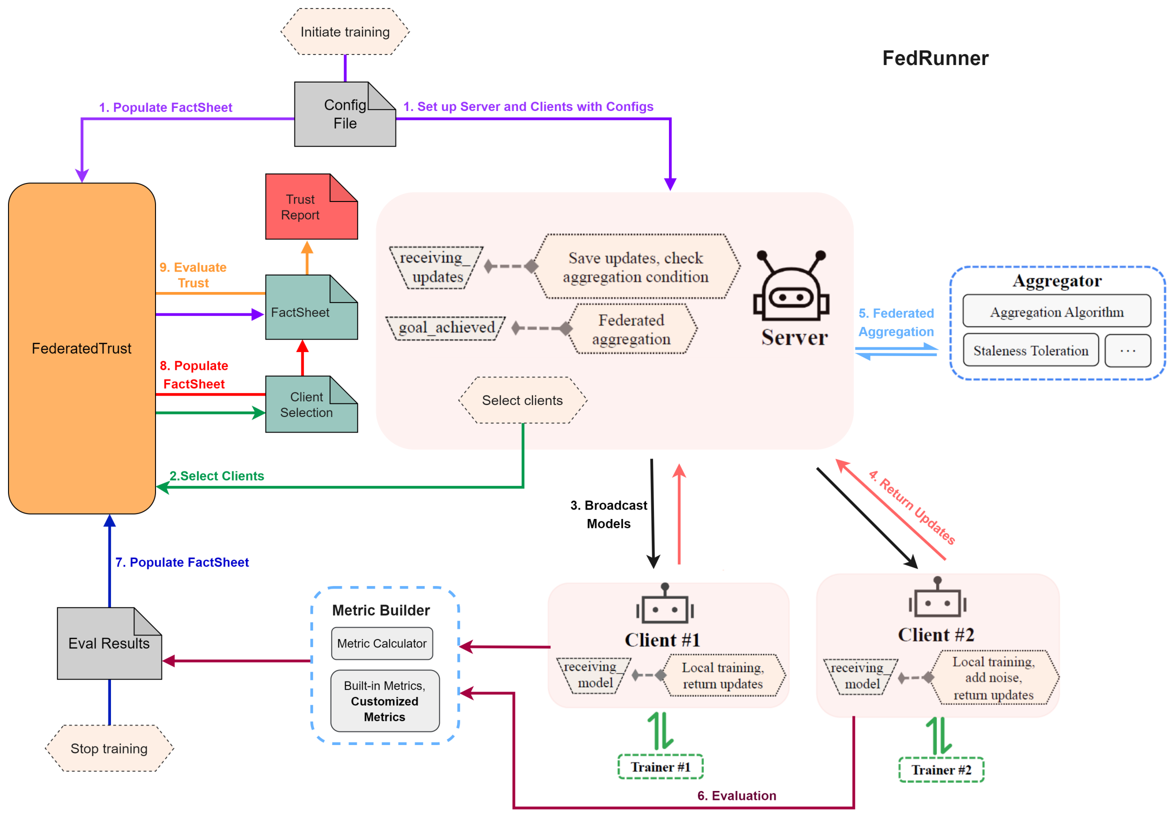


Figure 4.2: Overview of the interaction between FederatedScope with FederatedTrust

The explanations of the components and interaction steps are as follows:

1. The FedRunner initiates the training process and sets up the sever and the clients with the configurations from the config file and at the same time populates the FactSheet with the config file through *FederatedTrust*.
2. The server selects the specified percentage of clients and generates or updates the "Client Selection" file through *FederatedTrust*.
3. The server broadcasts the current global model to the selected clients.
4. Selected clients receive the current model and starts local training. When training is finished, each client sends their model updates back go the server.
5. The server calls the Aggregator to perform federated aggregation over all the received model updates.
6. After each round of training, clients receive the message from server (not shown on the figure) to perform model evaluation. Clients call the metric builder to perform metric calculations. The evaluation results are written into the "Eval Results" file.
7. After the final round of training, the FedRunner stops the training process and populates the aggregated evaluation results to the FactSheet through *FederatedTrust*.
8. *FederatedTrust* the populates FactSheet with the final client selection frequency map from the "Client Selection" file.
9. *FederatedTrust* calls the evaluate function to evaluate a trustworthiness score from the FactSheet and generates trustworthiness report in the output directory of the FederatedScope framework.

* Note that the both the files in gray color - "Config" file and "Eval Results" - are documents generated by the FederatedScope framework originally. Additional custom metrics are added under the Metric Builder.

Algorithm Pseudocode

The detailed step-by-step execution of the training of an FL model in FederatedScope using *FederatedTrust* as a third party library to evaluate the trustworthiness level is summarized in Algorithm 1.

Algorithm 1 Training in FederatedScope with FederatedTrust

```
Input: $N$ clients, sampling size $m$, a central server $S$, total number of iterations $T$,
initial model $\bar{w}^{(0)}$, setup configurations $C$, FederatedTrust metric manager $ft$
Output: Evaluation results, trustworthiness report
1: $S$ sends the hashed ids of all clients $i \in [N]$ and $C$ to $ft$
2: $ft$ creates FactSheet with information from $C$
3: $ft$ creates a map of hashed client ids to values of 0 representing initial selection rate
4: $S$ sends the model meta data to $ft$
5: $S$ request class distribution information from all clients $i \in [N]$
for clients $i \in [N]$ do
    Client $i$ uses $ft$ function to calculate the sample size per class of local data
    $ft$ creates or updates the class distribution map of hashed labels to sample size
end for
for $t = 0$ to $T$ do
    $S$ randomly samples $D^{(t)} \subset [N]$ clients withe size of $m$
    $S$ sends the hashed ids of the selected clients to $ft$
    $ft$ updates the client selection rate map
    $S$ broadcasts the current model $\bar{w}^{(t)}$ to all clients $i \in D^{(t)}$
    for clients $i \in D^{(t)}$ do
        Client $i$ performs local training with $\bar{w}^{(t)}$
        Client $i$ sends new model updates $w_i^{(t+1)}$ back to $S$
    end for
    $S$ performs secure aggregation of all updates received into a new global model $\bar{w}^{(t+1)}$
end for
$S$ sends final global model $\bar{w}'$ to every client $i \in [N]$ for performance evaluation
for clients $i \in [N]$ do
    Client $i$ computes evaluation metrics with local test data and global model $\bar{w}'$
    Client $i$ sends the evaluation results back to $S$
end for
$S$ aggregates the evaluation results and sends them to $ft$
$ft$ receives the evaluation results and populates the FactSheet with them
$S$ asks $ft$ to evaluate the trustworthiness of the model
$ft$ computes the trustworthiness score and generates a report JSON and print message
```

4.2 Implementation

As discussed in the limitation sections under each pillar in Chapter 3, not all metrics have feasible methods of calculation in a non-productionalized research environment. For the implementation of *FederatedTrust* in this work, the strategy is to build a minimum viable product, Prototype v1, with the basic features and metrics that can be calculated in the FederatedScope framework. Prototype v1 is a proof-of-concept algorithm showing how a unified way of calculating the trustworthiness score of FL models can work in an existing FL framework based on a well-researched taxonomy. Future versions of the prototype can be developed based on v1 by incorporating all the metrics from the full taxonomy as seen in Fig 3.1.

4.2.1 Taxonomy

Figure 4.3 shows a reduced taxonomy tailored for Prototype v1. There are still six pillars of trustworthy FL in the reduced taxonomy:

- Robustness - Privacy - Fairness - Explainability - Accountability

- Architectural Soundness

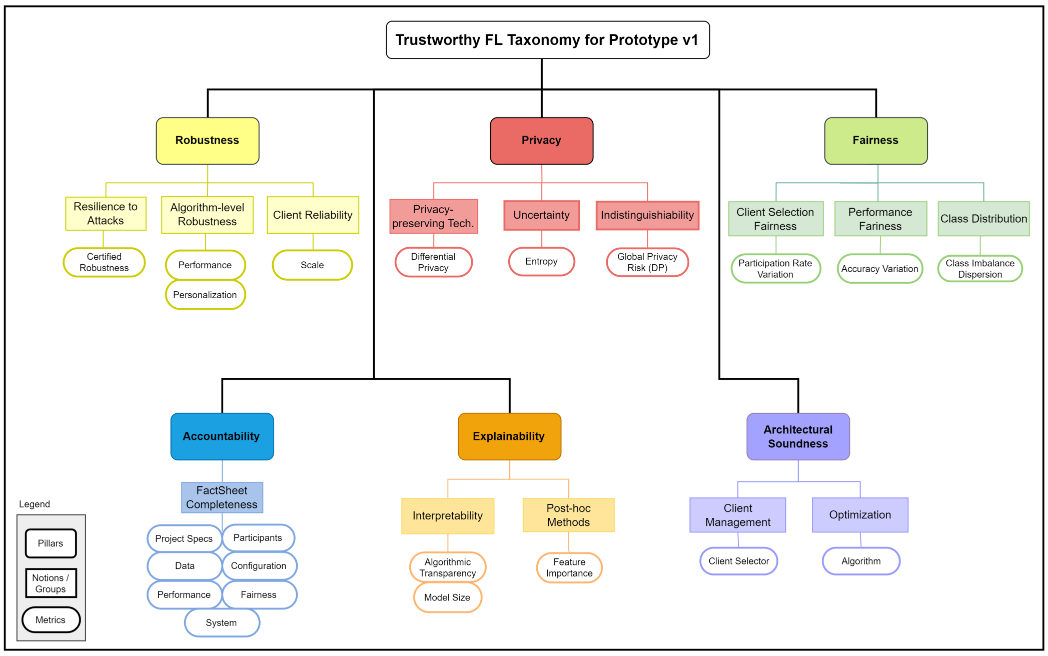


Figure 4.3: Reduced Taxonomy for Prototype v1

The list of omitted notions and metrics and the reasons for not including them in Prototype v1 are:

- Poisoning Defense: The use of poisoning defense is often integrated in the algorithm, for example, robust aggregation. It is difficult to quantify the usage of poisoning defense mechanism unless the information is documented. However, another way to verify this metric could be checking the empirical or the certified robustness against poisoning attacks.

- System-level Robustness: Neither testing data nor error rates are readily available in the FL framework.
- Client Reputation: There is no clear way to measure client reputation in a simulated environment. It requires more inspections to the client data provenance and this kind of inspection is not easily quantifiable.
- Data Quality: The proposed calculation for this metric needs to be executed during every round of training which creates a high computation overhead for the FL model training process.
- Privacy Preserving Techniques: One representative technique is selected as a metric because the use of encryption-based or anonymization-based techniques is not documented in the configuration file.
- Information Leakage Risk: The proposed calculation for this metric needs to be executed during every round of training by running extra neural networks which would incur computation overhead.
- Discrimination Index: This metric requires the knowledge of one protected sensitive attribute. It is unclear how this kind of calculation can be carried out in practice without the sharing of sensitive information among clients and the server.
- Monitoring: Auditing information is not readily available in a simulated environment.
- Model Management: Versioning and model replacement trigger information are not readily available.

4.2.2 Metric Definitions

This section contains six tables of metrics for each pillar as seen in the reduced taxonomy (Figure 4.3). Each metric has the following properties:

- **Metric**: name of the metric, as seen in Chapter 3
- **Description**: definition of metric and examples
- **Input**: input document and/or input data needed to calculate the metric
- **Output**: raw output of the metric before normalization in one of the formats below:
  - 0/1 - [m,n] - % - Integer - Real
  where [m,n] means that it is a range from m to n.
- **Dependency**: dependent elements or pre-conditions required for the computation

Robustness				
Metric	**Description**	**Input**	**Output**	**Dependency**
Certified Robustness	Minimum perturbation required to change the classification	Test sample, Model, Number of classes	Real	Model is neural network
Performance	Test accuracy of the global model	Model, Test data	%	Metric Builder and FactSheet
Personalization	Use of personalized FL techniques	FactSheet	0/1	Can be found in Config file
Scale	Number of clients representing the scale of the project	FactSheet	Integer	Can be found in Config file

Table 4.1: Metrics for Robustness

Privacy				
Metric	**Description**	**Input**	**Output**	**Dependency**
Differential Privacy	Use of global or local DP as a privacy defense	FactSheet	0/1	Can be found in FactSheet
Entropy	Uncertainty in predicting the value of a random variable	Number of clients	[0, 1]	Number of clients can be found in Config file
Global Privacy Risk	Maximum privacy risk with DP based on ϵ	ϵ in FactSheet	%	DP is used and ϵ value can be found in Config file

Table 4.2: Metrics for Privacy

Fairness				
Metric	**Description**	**Input**	**Output**	**Dependency**
Participation Rate Variation	Uniformity of distribution of participation rate among clients	Client selector info	[0, 1]	Client Selection file
Accuracy Variation	Uniformity of distribution of performance among clients	Model, Test data	[0, 1]	Metric Builder and FactSheet
Class Imbalance	Average class imbalance estimation among clients	Training data, labels	[0, 1]	Secure aggregation

Table 4.3: Metrics for Fairness

Explainability				
Metric	**Description**	**Input**	**Output**	**Dependency**
Algorithmic Transparency	Interpretability of the model by design	Algorithm name, Score map	[1, 5]	Algorithm name to score map is defined
Model Size	Number of featurs/Depth of decision tree/Number of trainable parameters in neural networks	FactSheet	Integer	Model meta data
Feature Importance	Average variance of feature importance scores	Model, test sample, batch size	[0, 1]	Test sample data

Table 4.4: Metrics for Explainability

Accountability				
Metric	**Description**	**Input**	**Output**	**Dependency**
Project Specs	Existence of content in the FactSheet	FactSheet	0/1	Correct FactSheet template
Participants	Existence of content in the FactSheet	FactSheet	0/1	Correct FactSheet template
Data	Existence of content in the FactSheet	FactSheet	0/1	Correct FactSheet template
Configuration	Existence of content in the FactSheet	FactSheet	0/1	Correct FactSheet template
Performance	Existence of content in the FactSheet	FactSheet	0/1	Correct FactSheet template
Fairness	Existence of content in the FactSheet	FactSheet	0/1	Correct FactSheet template
System	Existence of content in the FactSheet	FactSheet	0/1	Correct FactSheet template

Table 4.5: Metrics for Accountability

Architectural Soundness				
Metric	**Description**	**Input**	**Output**	**Dependency**
Client Selector	Use of a client selector scheme other than random selection	FactSheet	0/1	Can be found in FactSheet
Optimization Algorithm	Choice of optimization algorithm	FactSheet, Score map	%	Algorithm benchmarks are available and reliable

Table 4.6: Metrics for Architectural Soundness

4.2.3 FederatedTrust Algorithm

As seen in the list of metric definitions in Table 4.1, 4.2, 4.3, 4.4, 4.5 and 4.6, the FactSheet is the input for a majority of the metric calculations. The algorithm was implemented based on the three following concepts:

1. Consolidate as much facts about the FL model as possible into the FactSheet and use it as the major source of input for metric calculations
2. Create a generic pillar class that can be used by every type of pillar to reduce code repetition since the structure and the main functional requirement of the metric calculation (FR3 and FR4) apply to all pillars
3. To enable the flexibility for point 2, use a configurable approach for the metric definition so that by simply defining the input, output and calculation operation function new metrics can be added to the evaluation algorithm

Technology Stack

The *FederatedTrust* algorithm was implemented as a Python package since most FL frameworks had been developed using Python libraries. The implemented solution does not run the FL model but in the unit tests PyTorch library is used because that is the ML library used in the FederatedScope framework. The statistical calculations in the code are done by using the numpy, scipy and sklearn libraries.

Inputs and Configurations

1. FactSheet:

Code listing 4.1 shows part of the FactSheet template that is built on the IBM Accountable FL FactSheet example [85]. This template is to be populated with facts from the configuration file and the evaluation results from the FederatedScope framework. It is a JSON file containing seven sections of facts describing an FL model:

- Project Specification: overview, purpose, background
- Data: provenance, pre-processing
- Participants: number of clients, sample client rate, client selector
- Configuration: optimization algorithm, training model, personalization, differential privacy, hyperparameters
- Performance: test loss, test accuracy, test feature importance CV, CLEVER score
- Fairness: test accuracy CV, selection CV, class imbalance CV
- System: average training time, average model size, average upload bytes, average download bytes

```
{
  "project": {
    "overview": "",
    "purpose": "",
    "background": ""
  },
  "data": {
    "provenance": "",
    "preprocessing": ""
  },
  "participants": {
    "client_num": 0,
    "sample_client_rate": 0,
    "client_selector": ""
  },
  "configuration": {
    "optimization_algorithm": "",
    "training_model": "",
    "personalization": false,
    "differential_privacy": false,
    "dp_epsilon": 0,
    "trainable_param_num": 0,
    "total_round_num": 0,
```

Listing 4.1: FactSheet Template

2. Configuration File:

The following is an example of the configuration file used in the FederatedScope framework. It details the federated learning setting with client numbers, total round of training, model name, learning rate, etc. This configuration files is fed into the *FederatedTrust* algorithm to populate the FactSheet.

```
seed: 34567
expname: exp_1
federate:
  mode: standalone
  client_num: 50
  total_round_num: 25
  sample_client_num: 30
  sample_client_rate: 0.6
  join_in_info: ["num_sample"]
data:
  root: data/
  type: femnist
  splits: [0.6,0.2,0.2]
  batch_size: 10
  subsample: 0.05
  num_workers: 0
  transform: [['ToTensor'], ['Normalize', {'mean': [0.1307], 'std':
    [0.3081]}]]
model:
  type: convnet2
  hidden: 2048
  out_channels: 62
train:
  local_update_steps: 1
  batch_or_epoch: epoch
  optimizer:
    lr: 0.01
    weight_decay: 0.0
nbafl:
  use: True
  mu: 0.1
  constant: 1
  w_clip: 0.1
  epsilon: 20
grad:
  grad_clip: 5.0
criterion:
  type: CrossEntropyLoss
trainer:
  type: cvtrainer
eval:
  freq: 10
  metrics: ['acc', 'correct']
```

3. Class Distribution Map:

An example of the class distribution map is shown in the code listing below. The hashed IDs represent either the labels or classes, and the values represent the sample size per class.

```
{
  "oj": 688,
  "m0": 674,
  "v2m": 62,
  "nR": 626,
  "q2": 680,
  "p2": 709,
  "BBX": 50,
  "k5": 690,
  "MA": 54,
 ...
```

4. Client Selection Map:

The following is an example of the client selection map. The hashed IDs represent the client IDs and the values are the current selection or participation rate at a particular round of training.

```
{
  "jR": 0.46,
  "k5": 0.52,
  "l5": 0.58,
  "m0": 0.66,
  "nR": 0.602,
  "oj": 0.40,
  "p2": 0.582,
  "q2": 0.50,
  "rE": 0.68,
  ...
```

5. Metric Configuration:

The metric configurations are stored in a JSON format and the metric object structure is illustrated in Figure 4.4.

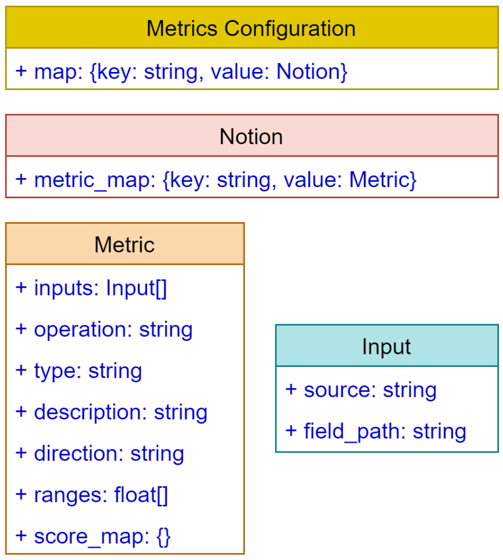


Figure 4.4: Metric Object Structure

The input object contains the source of the input data and the field path which is the reference of the input data within the source. For example, the input data for the number of clients is sourced from the "FactSheet", and the field path of the data is "participants\client_num".

Overall, there are five types of metrics:

- **true_score:** the input value directly reflects the output score
- **ranges:** the defined range where the input value falls in reflects the output score
- **score_mapping:** the input value is mapped to another value as the output score
- **score_ranking:** the input value is mapped to a ranking as the output score
- **property_check:** the input value should be be present

Code listing 4.2 shows two examples of true_score metrics. For example, under the algorithm robustness notion, there are two metrics: performance and personalization. Both metrics have input values from the FactSheet which can be directly used in the trustworthiness score calculation without any post-processing.

```
  "performance": {
    "inputs": [
      {
        "source": "factsheet",
        "field_path": "performance/test_acc_avg"
      }
    ],
    "operation": "get_value",
    "type": "true_score",
    "description": "Performance of the model."
  },
  "personalization": {
    "inputs": [
      {
        "source": "factsheet",
        "field_path": "configuration/personalization"
      }
    ],
    "operation": "get_value",
    "type": "true_score",
    "description": "Use of personalized FL techniques."
  }
```

Listing 4.2: True Score Metrics

Code listing 4.3 shows an example of a metric that calculates based on ranges. For example, the number of clients which reflects the scale of the FL model can fall in the ranges [0, 100], [10, 100], [100, 1000] and so on. and Each range reflects a score. In this case, the smaller the scale, the higher the client reliability so that the lower ranges have better trustworthiness scores than the higher ranges.

```
"client_reliability": {
  "scale": {
    "inputs": [{
      "source": "factsheet",
      "field_path": "participants/client_num"
    }],
    "operation": "get_value",
    "type": "ranges",
    "direction": "desc",
    "ranges":[10,100,1000,10000,100000,1000000,10000000],
    "description": "Scale of the project."
  }
}
```

Listing 4.3: Range Metric

Code listing 4.4 shows an example of a metric that uses score mapping. This algorithmic transparency metric is a direct reference from the previous work of Trusted AI algorithm [49]. In this example, every type of ML/DL model is assigned a score from 1 to 5 representing their level of algorithmic transparency as suggested in the work by Arrieta et al. [42].

```
"algorithmic_transparency": {
    "inputs": [{
      "source": "factsheet",
      "field_path": "configuration/training_model"
    }],
    "operation": "get_value",
    "type": "score_mapping",
    "score_map": {
      "RandomForestClassifier": 4,
      "KNeighborsClassifier": 3,
      "SVC": 2,
      "GaussianProcessClassifier": 3,
      "DecisionTreeClassifier": 5,
      "MLPClassifier": 1,
      "AdaBoostClassifier": 3,
      "GaussianNB": 3.5,
      "QuadraticDiscriminantAnalysis": 3,
      "LogisticRegression": 4,
      "LinearRegression": 3.5,
      "Sequential": 1,
      "CNN": 1
    },
    "description": "Mapping of Learning techniques to the level of
   transparency."
}
```

Listing 4.4: Score Mapping Metric

Code listing 4.5 shows an example of metric that checks whether a property exists or not. This example is used for checking if the client_selector property has any value in the FactSheet.

```
"client_management": {
  "client_selector": {
    "inputs": [{
      "source": "factsheet",
      "field_path": "participants/client_selector"
    }],
    "operation": "get_value",
    "type": "property_check",
    "description": "Use of a non random client selector scheme."
  }
}
```

Listing 4.5: Property Check Metric

Metric Operations

For metric categories that cannot directly use the input value as the output score, additional calculation operations need to be performed. In this section, each metric operation is explained regarding their functionality, code and the sample usage for a metric in the Taxonomy.

Operation 1: check_properties

This operation maps each argument to 0 for undefined value or 1 for defined value. The mean of the resultant list of binary values computes the percentage of value definitions among the all arguments. This is used for calculating the metrics under the FactSheet completeness notion of the Accountability pillar. For example, in the FactSheet, the project specifications metrics is calculated by checking if the overview, purpose and background input values are defined in the FactSheet.

```python
def check_properties(*args):
    result = map(lambda x: x is not None and x != "", args)
    return np.mean(list(result))
```

Operation 2: get_entropy

This operation is to calculate the entropy of a dataset. In this calculation, n represents the number of samples and the assumption is that every sample has an equal probability of being identified. Therefore, this is a simplified way to evaluate the entropy. This is used to calculate the entropy metric under the privacy pillar.

```python
def get_entropy(n):
    entropy = -1 * np.sum(np.log2(1/n)*(1/n))
    return entropy
```

Operation 3: get_global_privacy_risk

This operation calculates the maximum privacy risk in a model that uses DP based on the ϵ value. The mathematical equation can be found in section 3.2.4.

```python
def get_global_privacy_risk(dp, epsilon, n):
    if dp is True and isinstance(epsilon, numbers.Number):
        return 1 / (1 + (n - 1) * math.pow(e, -epsilon))
    else:
        return 1
```

Operation 4: get_cv

The CV is a standardized measure of dispersion of a probability distribution. It calculates the ratio of the standard deviation to the mean and shows the extent of variability of data in a sample in relation to the mean of the population. This is used to calculate all the metrics under privacy pillar and is part of the calculation for the feature importance metric under the explainability pillar.

```python
def get_cv(std, avg):
    return std / avg
```

Operation 5: get_feature_importance

The feature importance is calculated by the Shapley value. The Shapley value of of one specific feature is the average marginal contribution of this feature across all possible feature combinations. The algorithm can be seen in appendix A.2. The DeepExplainer function of the shap library in Python provides approximate SHAP values for deep learning models. This is used to calculate the feature importance metric under the explainability pillar. The dispersion of the feature importance scores is calculated to show how vaired the importance scores are. The more dispersed the feature importance scores, the higher the chance that the model is more explainable.

```python
import shap

def get_feature_importance_cv(test_sample, model, cfg):
    cv = 0
    batch_size = cfg['batch_size']
    device = cfg['device']
    if isinstance(model, torch.nn.Module):
        batched_data, _ = test_sample

        n = batch_size
        m = math.floor(0.8 * n)

        background = batched_data[:m].to(device)
        test_data = batched_data[m:n].to(device)

        e = shap.DeepExplainer(model, background)
        shap_values = e.shap_values(test_data)
        sums = np.array([shap_values[i].sum() for i in range(len(
    shap_values))])
        abs_sums = np.absolute(sums)
        cv = variation(abs_sums)
    return cv
```

Operation 6: get_clever_score

The CLEVER score is calculated by using the ART Python library. This operation takes a test sample, the global model and configurations like the batch size, number of classes, and the learning rate as parameters. The output is the CLEVER score for an untargeted attack, which represents the lower bound of perturbations required for the adversary to succeed in changing the classifier.

```python
from art.estimators.classification import PyTorchClassifier
from art.metrics import clever_u

def get_clever_score(test_sample, model, cfg):
    nb_classes = cfg['nb_classes']
    lr = cfg['lr']
    images, _ = test_sample
    background = images[-1]

    criterion = nn.CrossEntropyLoss()
    optimizer = optim.Adam(model.parameters(), lr)

    # Create the ART classifier
    classifier = PyTorchClassifier(
        model=model,
        clip_values=(0.0, 255.0),
        loss=criterion,
        optimizer=optimizer,
        input_shape=(1, 28, 28),
        nb_classes=nb_classes,
    )
    score_untargeted = clever_u(classifier,
                                background.numpy(),
                                10,
                                5,
                                R_L2,
                                norm=2,
                                pool_factor=3,
                                verbose=False)
    return score_untargeted
```

Scoring Functions

The output value of the metric operations come in different formats as indicated in the metric definitions (4.2.2). In order to combine these values into an understandable trustworthiness score, different scoring functions are used to translate the output values into a normalised score between [0, 1]. The logic of each scoring function is explained below.

Scoring Function 1: get_range_score

This function takes an input value, an array of range boundaries and a direction enum as parameters to calculate the index of the range that the input value corresponds to. The index and the total number of ranges are then used to calculate a normalized score. The direction can be specified in the metric definition: “asc” means that the ranges at the higher indices in the ranges array has higher scores than those at the lower indices; “desc” means the opposite holds. This is used to evaluate the scale metric under the robustness pillar, the entropy metric under the privacy pillar, all metrics under the fairness pillar, and the model size and the feature importance metrics in the explainability pillar.

```python
def get_range_score(value, ranges, direction):
    score = 0

    if ranges is None:
        logger.warning("Score ranges are missing")
    else:
        total_bins = len(ranges) + 1
        bin = np.digitize(value, ranges, right=True)
        if direction == 'desc':
            score = 1 - (bin / total_bins)
        else:
            score = bin / total_bins

    return score
```

Scoring Function 2: get_normalized_score

This function takes a key and a defined score map as parameters to calculate the normalized mapped score for the given key. This is used to evaluate the algorithmic transparency metric under the explainability pillar.

```python
from sklearn import preprocessing

def get_normalized_score(score_key, score_map):
    score = 0;

    if score_map is None:
        logger.warning("Score map is missing")
    else:
        keys = [key for key, value in score_map.items()]
        scores = np.array([value for key, value in score_map.items()])
        normalized_scores = preprocessing.normalize([scores])
        normalized_score_map = dict(zip(keys, normalized_scores[0]))
        score = normalized_score_map.get(score_key, np.nan)

    return score
```

Scoring Function 3: get_ranked_score

This function takes a key, a defined score map and a direction enum as parameters to calculate a ranked score. The score map is first sorted by the score values by the direction specified. Based on the sorted array, the key is used to find the position of the mapped score in the sorted array. The index and the length of the score array is used to calculated a normalized ranking for the key in the score map. This is used to calculate the algorithm metric under the Architectural Soundness pillar.

```python
def get_ranked_score(score_key, score_map, direction):
    score = 0

    if score_map is None:
        logger.warning("Score map is missing")
    else:
        sorted_scores = sorted(score_map.items(),
                               key=lambda item: item[1],
                               reverse=direction == 'desc')

        sorted_score_map = dict(sorted_scores)

        for index, key in enumerate(sorted_score_map):
            if key == score_key:
                score = (index + 1) / len(sorted_score_map)

    return score
```

Metric Scoring System

The previous section 4.2.3 describes the functional calculation steps to evaluate a metric from the necessary input data to an output value by the metric definition. For example, how to calculate the coefficient of variation or the global privacy risk from raw input values of test accuracy or the ϵ. In order to present an understandable trustworthiness level, a score within the range of [0, 1] or in other words 0-100%, is aggregated from all the metrics. Section 4.2.3 presents the utility functions available to arrive at a normalized trust score. However, the logic for comparison is missing and this section fills the gap with the design behind the scoring system for each metric based on the analysis in Chapter 3. As illustrated in section 4.2.3, each Metric object has a type, a direction, optional ranges and score mappings. The direction, ranges and score mappings are used in the scoring functions to calculate the trust score based on the metric value.

1. Robustness Pillar

    - **certified robustness:**
      the CLEVER score falls in one of the ranges below, the higher the range the higher the robustness level

      ```
      [0, 0.2, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0, 2.2, 2.4,
          2.6, 2.8, 3.2, 3.4, 3.6, 3.8, 4.0]
      ```

    - **performance:** value is percentage and therefore is bounded by [0, 1]
    - **personalization:** True or False values correspond to 1 or 0
    - **scale:** the lower the number of clients in the range [10^1, 10^{10}], the higher the higher the robustness level

2. Privacy Pillar

    - **differential privacy:** True or False values correspond to 1 or 0
    - **entropy:** value is bounded by [0, 1], the higher the entropy the higher the privacy level
    - **global privacy risk:** value is bounded by [0, 1], the lower the risk the higher the privacy level

3. Fairness Pillar

    - **selection variation:** value is bounded by [0, 1], the lower the variation the higher the fairness level
    - **accuracy variation:** value is bounded by [0, 1], the lower the variation the higher the fairness level

- **class imbalance:** value is bounded by [0, 1], the lower the imbalance the higher the fairness level

4. Explainability Pillar

- **algorithmic transparency:**
The score mapping from the previous work [49] based on the level transparency is used. The higher the score, the higher the transparency level.

```
    "score_map": {
      "RandomForestClassifier": 4,
      "KNeighborsClassifier": 3,
      "SVC": 2,
      "GaussianProcessClassifier": 3,
      "DecisionTreeClassifier": 5,
      "MLPClassifier": 1,
      "AdaBoostClassifier": 3,
      "GaussianNB": 3.5,
      "QuadraticDiscriminantAnalysis": 3,
      "LogisticRegression": 4,
      "LinearRegression": 3.5,
      "Sequential": 1,
      "CNN": 1
    }

```

- **model size:**
The number of features (i.e. trainable parameters in NN) falls in one of the ranges below. The lower the range, the higher the interpretability level.

```
    [10, 50, 100, 500, 1000, 5000, 10000, 50000, 100000]

```

- **feature importance variation:** value is bounded by [0, 1], the higher the variation the higher the explainability level of the features

5. Accountability Pillar

- **project specifications:** existence of information corresponds to 1, otherwise 0
- **participants:** existence of information corresponds to 1, otherwise 0
- **data:** existence of information corresponds to 1, otherwise 0
- **configuration:** existence of information corresponds to 1, otherwise 0
- **performance:** existence of information corresponds to 1, otherwise 0
- **fairness:** existence of information corresponds to 1, otherwise 0
- **system:** existence of information corresponds to 1, otherwise 0

6. Architectural Soundness Pillar

- **client selector:** use of a selector scheme corresponds to 1, otherwise 0
- **optimization algorithm:**
  The following benchmarking of FL algorithms provided by FederatedScope is used as a ranking system for the algorithms. The higher the benchmarked performance, the higher ranked the is the algorithm. However, if resources allow, it is more advisable to test run each of the FL algorithm on the intended federated client data to produce more accurate benchmarking information pertaining for the FL data and task of interest.

```
    "score_map": {
      "FedAvg": 0.8493,
      "FedOpt": 0.8492,
      "FedProx": 0.8477,
      "FedBN": 0.8548,
      "pFedMe": 0.8765,
      "Ditto": 0.8661,
      "FedEM": 0.8479
    }

```

Algorithm Structure

Figure 4.5 shows the code structure on the left and the UML diagrams for the *TrustMetricManager* and the *TrustPillar* classes on the right.

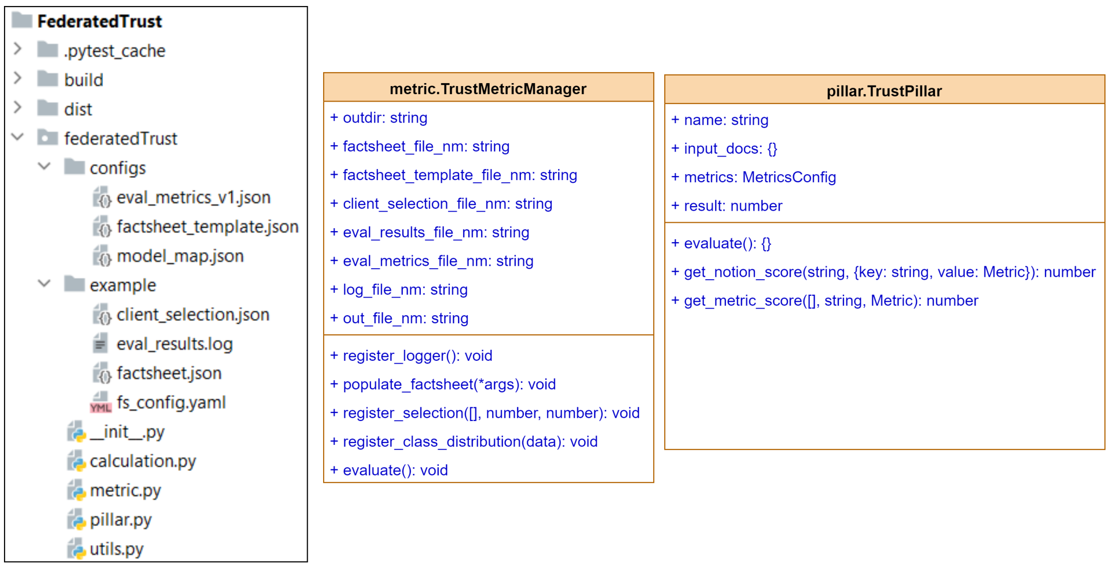

FederatedTrust
.pytest_cache
build
dist
federatedTrust
configs
eval_metrics_v1.json
factsheet_template.json
model_map.json
example
client_selection.json
eval_results.log
factsheet.json
fs_config.yaml
__init__.py
calculation.py
metric.py
pillar.py
utils.py
metric.TrustMetricManager
+ outdir: string
+ factsheet_file_nm: string
+ factsheet_template_file_nm: string
+ client_selection_file_nm: string
+ eval_results_file_nm: string
+ eval_metrics_file_nm: string
+ log_file_nm: string
+ out_file_nm: string
+ register_logger(): void
+ populate_factsheet(*args): void
+ register_selection([], number, number): void
+ register_class_distribution(data): void
+ evaluate(): void
pillar.TrustPillar
+ name: string
+ input_docs: {}
+ metrics: MetricsConfig
+ result: number
+ evaluate(): {}
+ get_notion_score(string, {key: string, value: Metric}): number
+ get_metric_score([], string, Metric): number

Figure 4.5: Code Structure and UML Class Diagram

As seen in the code structure, the configs folder holds all the configuration files such as the initial FactSheet template (factsheet_template.json) and the metric definitions JSON (eval_metrics_v1.json). The example folder holds the example configuration file (fs_config.log) and evaluation report (eval_results.log) from the FederatedScope framework. Examples of a populated FactSheet document and the generated client selection map (client_selection.json) are also provided.

The core functionalities of the algorithm are written in the *pillar.py* and *metrics.py* modules. The *TrustPillar* class defines the pillar objects that are deserialized from the metric definitions JSON. The *TrustMetricManager* serves as an interface object between the parent FL framework and the trustworthiness evaluation algorithm. The *calculation.py* module contains all the metric operations (4.2.3) and scoring functions (4.2.3), and the *utils.py* module contains utility functions used for file I/O operations.

Figure 4.6 below shows the design of the FederatedTrust algorithm. A FactSheet is first generated and populated in the process as illustrated in section 4.1.5. The inputs from the FactSheet are evaluated against the corresponding metrics by performing metric operations and scoring functions to arrive at a normalized trust score. The metric scores under each notion are then aggregated based on their weights. The notion trust scores are then aggregated into the pillar scores. The final trust score of the model is a weighted average of the pillar scores.

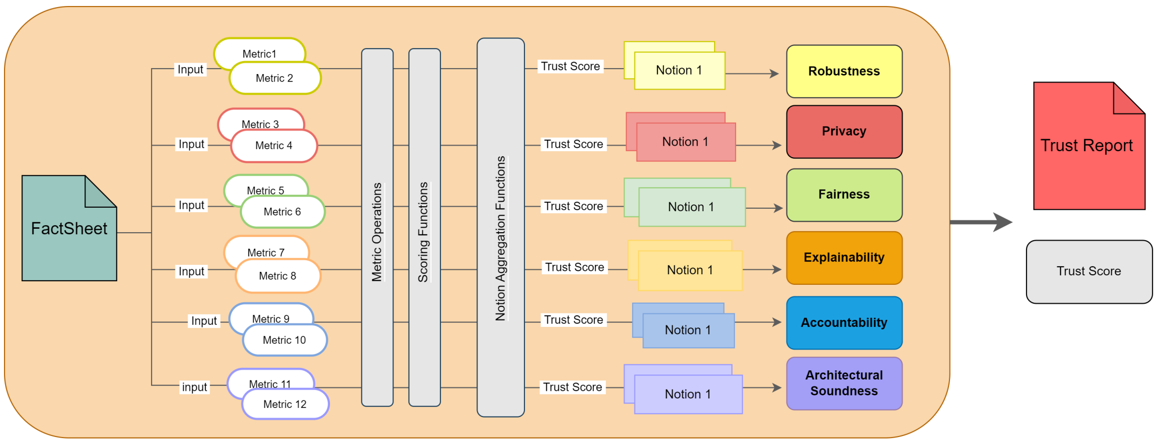


Figure 4.6: Design for the FederatedTrust Algorithm

4.3 Deployment

To demonstrate the usefulness of the proposed *FederatedTrust* algorithm, Prototype v1 was developed and built as a local Python package and was tested in the FederatedScope framework as a proof-of-concept. The following section details the steps to build and import the *FederatedTrust* library in any FL development framework.

Installation Guide

1. The following Python packing libraries are the prerequisites for building the *FederatedTrust* package locally.

```
> pip install wheel
> pip install setuptools
> pip install twine
```

2. Navigate to the *FederatedTrust* directory and run the following command to build the package.

```
..\FederatedTrust> python setup.py bdist_wheel
```

3. Within the FederatedScope framework, install the *FederatedTrust* package locally.

```
..\FederatedScope> pip install ..\FederatedTrust-0.1.0-py3-none-any.whl
```

4. Import the TrustMetricManager and initiates instance with a required output directory.

```python
from federatedTrust.metric import TrustMetricManager

# set up TrustMetricManager
# a output directory path is required
trust_metric_manager = TrustMetricManager(output_dir)
```

5. Example uses of the TrustMetricManager instance

```python
# populates the FactSheet with the configuartion file
trust_metric_manager.populate_factsheet(cfg_file="config.yaml")

# populates the FactSheet with the evaluation results file
trust_metric_manager.populate_factsheet(eval_results_file="eval_results.
    log")

# updates the client selection frequency map
trust_metric_manager.register_selection(clients, total_round_number, -1)

# updates the unified class distribution with the training data on the
    client side
trust_metric_manager.register_class_distribution(data)

# evaluates the trustworthiness level of the given model
trust_metric_manager.evaluate(test_sample, model, cfg)
```

4.3.1 Use Case

FederatedScope framework offers a range of datasets, machine learning models and FL algorithms. For testing the deployment of the trustworthiness evaluation algorithm, a basic use case in the framework is chosen and four different sets of environmental configurations were tested and compared. Table 4.7 shows the basic configurations for the use case. The use case runs an FL training in standalone mode with clients and one central server over a defined number of iterations. The optimization algorithm is FedAvg and the training model is a convolutional neural network, CovNet2. The federated datasets that the clients hold locally are from the FEMNIST image dataset of handwritten digits and letters. The dataset has 62 different classes (10 digits, 26 lowercase, 26 uppercase) and the images are 28 by 28 pixels. The client selector strategy is random sampling based on a defined percentage.

Mode	Standalone
Algorithm	FedAvg
Dataset	FEMNIST
Trainer Model	CovNet2

Table 4.7: Use case Basic Configurations

Evaluation

Experiment 1: The experiment set up 10 clients for the federation, with 50% sampling rate. The training ran 5 rounds to produce a an aggregated global mode. This experiment did not use personalization techniques therefore all clients shared the same global model. The expected result was that performance and fairness levels would be very low due to few number of rounds of training and low sampling rate. The privacy protection level was expected to be low because no DP was added.

Client Number	10
Sample Rate	20%
Rounds	5
Personalization	No
Differential Privacy	No

Table 4.8: Experiment 1 Setting

Experiment 2: 50 clients were set up for the federation. In every iteration, the server randomly sampled 60% of the clients to participate in the training. The experiment ran 25 rounds of iterations. No personalization techniques were used but DP was added by switching on the NbAFL approach. The expected performance and fairness levels would be higher due to higher sampling rate and higher number of training rounds. The privacy protection level was expected to be low because the DP ϵ was set as 20.

Client Number	50
Sample Rate	60 %
Rounds	25
Personalization	No
Differential Privacy	$\epsilon = 20$

Table 4.9: Experiment 2 Setting

Experiment 3: Similar configurations as Experiment 2 were used. The difference was in the ϵ value used for DP. The expected privacy protection would be higher.

Client Number	50
Sample Rate	60%
Rounds	25
Personalization	No
Differential Privacy	$\epsilon = 6$

Table 4.10: Experiment 3 Setting

Experiment 4: The experiment was medium scaled among all the experiment runs consisting of 100 clients with 40% sampling rate and 50 rounds of training. DP ϵ value was low at 6.

Client Number	100
Sample Rate	60%
Rounds	50
Personalization	No
Differential Privacy	$\epsilon = 6$

Table 4.11: Experiment 4 Setting

4.3.2 Result Discussion

For the four experiments, some variables are constant. Because of the same environment in FederatedScope framework, the same amount of meta data about the project specifications, data and participants can be extracted from all the models. For the project specification section, only the overview of the project could be inferred from the configuration file. The purpose and the background of the project were not specified in the model. The models all used a random sampling client selector so they all received the full 1 or 100% trust score for the client management notion. Since all experiments ran on the FedAvg aggregation algorithm, the optimization algorithm score was also the same for all experiments. The common results are shown in the Table 4.12 below. As of now, all metrics and notions have the same weights because the mechanism to determine the importance of each metric has not been analyzed and implemented.

Common Results in the Experiments

accountability	**0.9**		
factsheet completeness	0.9	project specs	0.33
		participants	1
		data	1
		configuration	1
		performance	1
		fairness	1
		system	1
architectural soundness	**0.78**		
client management	1	client selector	1
optimization	0.57	algorithm	0.57

Table 4.12: Common Results in the Experiments

Comparing Experiment 1 and 2

The following list of observations were made from the results of Experiment 1 and 2:

OB-1 The certified robustness score dropped from 0.48 to 0.19 from Experiment 1 to 2.

OB-2 The scale for both experiments have similar score since both have less than 50 clients.

OB-3 The privacy score increased from 0.31 to 0.64 from Experiment 1 to 2.

OB-4 The indistinguishability score is still 0 regardless of the use of DP.

OB-5 The fairness score increased from 0.25 to 0.47 with significant increase in selection fairness (from 0.08 to 0.83).

OB-6 The performance fairness dropped from 0.58 to 0.50 from Experiment 1 to 2.

OB-7 The explainability score increased from 0.61 to 0.73 with an increase for the feature importance score from 0.67 to 0.92.

trustworthiness score	**0.56**		
robustness	**0.5**		
resilience to attacks	0.48	certified robustness	0.48
algorithm robustness	0.035	performance personalization	0.07 0
client reliability	1	scale	1
privacy	**0.31**		
technique	0	differential privacy	0
uncertainty	0.92	entropy	0.92
indistinguishability	0	global privacy risk	0
fairness	**0.25**		
selection fairness	0.08	selection variation	0.08
performance fairness	0.58	performance variation	0.58
class distribution	0.08	class imbalance	0.08
explainability	**0.61**		
interpretability	0.55	algorithmic transparency model size	0.09 1
post-hoc methods	0.67	feature importance	0.67

Table 4.13: Experiment 1 - 10 clients, 20% sampling rate, 5 rounds, no DP

trustworthiness score	**0.65**		
robustness	**0.38**		
resilience to attacks	0.19	certified robustness	0.19
algorithm robustness	0.05	performance personalization	0.1 0
client reliability	0.91	scale	0.91
privacy	**0.64**		
technique	1	differential privacy	1
uncertainty	0.92	entropy	0.92
indistinguishability	0.0	global privacy risk	0.0
fairness	**0.47**		
selection fairness	0.83	selection variation	0.83
performance fairness	0.50	performance variation	0.50
class distribution	0.08	class imbalance	0.08
explainability	**0.73**		
interpretability	0.55	algorithmic transparency model size	0.09 1
post-hoc methods	0.92	feature importance	0.92

Table 4.14: Experiment 2 - 50 clients, 60% sampling rate, 25 rounds, DP with $\epsilon = 20$

The drop in certified robustness level in OB-1 could be related to the increase in the number of clients and the number of rounds. In theory, more aggregating parties provide more entries and surfaces for adversaries to insert backdoor perturbations for poisoning attacks. There is also higher chances for parties to collude when they are more in number. The higher number of rounds also mean that adversaries have more chances to administer attacks.

Even though the privacy score increased overall because of the use of DP in OB-3, the effectiveness of the privacy mechanism was not good because of the large value of ϵ chosen for the NbAFL approach. By relaxing the ϵ to a large value like 20, the privacy protection became lower because the larger the ϵ value the less noise was added to the data and therefore higher probability of identification by the adversary. The resultant indistinguishability score of 0 reflected the low privacy level in OB-4.

The significant increase for the fairness score in OB-5 was resulted from the overall increase in the number of clients, the client sampling rate and the number of rounds. The selection fairness improved with more clients participating and more training rounds, however, the drop in performance fairness in OB-6 could be due to the increased number of clients as well. More clients with different levels of heterogeneity in their data could influence the generalizability of the global model hence affecting the individual test accuracy at the client level. Furthermore, personalization techniques were not used so the global model were not be adapted to the clients.

Comparing Experiments 3 and 4

The following list of observations were made from the results of Experiment 3 and 4:

OB-8 The overall robustness score only increased from 0.36 to 0.38 from Experiment 3 to 4, even though there was a significant increase in the model performance from 0.09 to 0.36.

OB-9 The privacy score increased due to the increase in indistinguishability from Experiment 3 to 4.

OB-10 The client reliability score remained the same for both experiments.

OB-11 The feature importance score continued to increase to 0.92 in Experiment 4.

OB-12 The fairness score were almost the same for both experiments.

trustworthiness score	**0.64**		
robustness	**0.36**		
resilience to attacks	0.14	certified robustness	0.14
algorithm robustness	0.045	performance personalization	0.09 0
client reliability	0.91	scale	0.91
privacy	**0.68**		
technique	1	differential privacy	1
uncertainty	0.92	entropy	0.92
indistinguishability	0.11	global privacy risk	0.11
fairness	**0.47**		
selection fairness	0.83	selection variation	0.83
performance fairness	0.50	performance variation	0.50
class distribution	0.08	class imbalance	0.08
explainability	**0.65**		
interpretability	0.55	algorithmic transparency model size	0.09 1
post-hoc methods	0.75	feature importance	0.75

Table 4.15: Experiment 3 - 50 clients, 60% sampling rate, 20 rounds, DP with $\epsilon = 6$

trustworthiness score	**0.67**		
robustness	**0.38**		
resilience to attacks	0.05	certified robustness	0.05
algorithm robustness	0.18	performance personalization	0.36 0
client reliability	0.91	scale	0.91
privacy	**0.71**		
technique	1	differential privacy	1
uncertainty	0.92	entropy	0.92
indistinguishability	0.20	global privacy risk	0.20
fairness	**0.50**		
selection fairness	0.83	selection variation	0.83
performance fairness	0.58	performance variation	0.58
class distribution	0.08	class imbalance	0.08
explainability	**0.73**		
interpretability	0.55	algorithmic transparency model size	0.09 1
post-hoc methods	0.92	feature importance	0.92

Table 4.16: Experiment 4 - 100 clients, 60% sampling rate, 50 rounds, DP with $\epsilon = 6$

The improvement in the indistinguishablity score in OB-9 was caused by the increase in the number of clients. Assume random guessing was used, it was twice as difficult to guess the correct target among 100 clients as compared to 50 clients. The equal client reliability score in OB-10 was because the numbers of clients in both experiments fell within the range [50, 100] which gave the same score. The increase in feature importance score in OB-11 may be related to the increase in data samples from more clients and the better model performance. When the model learned the data better, the features had more impact on the prediction outcomes. The similar fairness score in OB-12 may be due to the fact that the ratio of between the increase in the number of clients and the increase in the number of rounds were the same while the sampling rate remained the same as well.

4.4 Limitations

As seen from the experiment results, there are several limitations to the *FederatedTrust* algorithm in terms of quantifying the trustworthiness level of FL models. On the one hand, some metrics are easily quantifiable and can represent a component of the trustworthiness level well. For example, the certified robustness is an attacker lower bound which is directly associated to resilience to attacks. On the other hand, some other metrics, like scale under client reliability, often have to be considered with other factors in order to represent the notion well. For example, the analysis in Chapter 3 shows that in practice , the individual client reputation level is an important factor for client reliability as well. However, it was difficult to quantify client reputation in the simulated experiment. Another example would be the selection fairness notion. Although the selection fairness metric was easily quantifiable by computing the dispersion of selection rate among the clients, this variation metric alone may not be the best representation of fairness for client selection in FL. Increasing the number to clients, the sample rate and the round of training could easily bring up the selection variation, but equality does not necessarily imply equity which is important for true fairness.

Another limitation of the algorithm is pertaining to the scoring system. In order to aggregate all the scores into a trustworthiness level between 0 and 1, all the metric values had to go through the metric operations and the scoring functions. The logic of the scoring functions has a high impact on how the trust score of each metric came out. Based on the pillar analysis in Chapter 3, there were a general directions of how every metric should impact the overall trustworthiness level, however, the concrete scoring maps and ranges were created based on subjective understanding and standards used from other studies. Their generalizability for other systems were not fully evaluated. Furthermore, the current weighing system for the metrics does not implement varied weights, meaning that every metric has the same weight under one notion and every notion has the same weight when aggregated into the final trustworthiness level. The weighing system could be enhanced by further analysis of the trade offs between pillars and metrics to produce a more balanced trustworthiness level of FL models.

Chapter 5

Summary and Conclusions

This thesis work first took a deep dive into the literature to study the existing work on Trustworthy AI and the state-of-the-art FL. During the survey and analysis of the existing Trustworthy AI pillars, namely robustness, privacy, fairness, explainability and accountability, comparisons to FL modes were made and the applicability of metric evaluations were considered. At the end of the analysis a comprehensive taxonomy describing the requirements for Trustworthy FL was created. A new pillar, Architectural Soundness, was added to the taxonomy as a special pillar for FL models to address the lack of trust brought by the complex distributed system. Furthermore, different notions and new metrics were added to each pillar specifically for the FL models, for example, system-level robustness, information leakage risk and client selection fairness. The limitations and drawbacks of individual metrics were also discussed. Based on the comprehensive taxonomy, a trustworthiness evaluation algorithm, named *FederatedTrust*, was designed with the goal to be a light-weight, configurable and flexible trust evaluation algorithm library. Before designing the algorithm, a list of existing state-of-the-art open-source FL development and benchmarking frameworks were assessed and one of the newest framework, FederatedScope, was selected as the reference FL framework to deploy and test the evaluation algorithm. The architecture of *FederatedTrust* and its interactions with the parent FL framework were designed and a list of functional requirements and privacy constraints were defined. The algorithm was implemented and tested in FederatedScope with the baseline FedAvg optimization algorithm on the FEMNIST dataset, A total of four experiments were ran with different FL settings varying the number of clients, rounds and differential privacy parameters. The generated experiment results, which were reports of the trustworthiness level of each components of the pillars, were compared and discussed. The trustworthiness report was able to report accurately the quantifiable metrics but less accurate for the notions or metrics that needed qualitative justifications as well. The experiments demonstrated the complexity of the task to quantify the trustworthiness level of FL models, and the *FederatedTrust* algorithm was the first attempt to a holistically assess an FL model based on a comprehensive trustworthiness taxonomy.

5.1 Future Work

Currently, the *FederatedTrust* evaluation algorithm only supports the FederatedScope framework. Because of the focus on FederatedScope framework, input variables were gathered mostly from the configuration file and the evaluation results generated by the FederatedScope framework itself. The design of the interactions between the framework and the *FederatedTrust* algorithm was also heavily based on how the framework conducted FL training processes. In the future, more FL development frameworks can be integrated and supported by further expanding the sources of inputs and the adaptability of the algorithm. Further more, the score aggregation step could be improved by performing more analysis on the trade offs between pillars and metrics to best handle the weight assignments. The scoring functions could also be improved by designing better score ranges and mappings. Furthermore, the current algorithm only supports computing trustworthiness scores of pillars and metrics either before or after training starts. Some metrics need to be calculated during every round of training, however, due to large computation overhead, conducting round-wise trust score evaluation with complex calculation methods was not feasible. There are still a number of metrics in the full taxonomy that were not implemented in the Prototpye v1 of the *FederatedTrust* algorithm. Potential future work can also focus on bringing a more realistic use case into the evaluation framework so that metrics that were omitted due to practical environmental constraints can be added. Lastly, further evaluation experiments should be conducted on different FL optimization algorithms and federated datasets for more comparative analyses.

Bibliography

[1] Y. Bengio, "Springtime for AI: The Rise of Deep Learning," 2016. [Online]. Available: https://www.scientificamerican.com/article/springtime-for-ai-the-rise-of-deep-learning/

[2] D. A. Ferrucci, "Introduction to "This is Watson"," *IBM Journal of Research and Development*, vol. 56, no. 3.4, pp. 1:1–1:15, 2012.

[3] J. D. Olga Russakovsky *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[4] D. Silver, A. Huang *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan 2016. [Online]. Available: https://doi.org/10.1038/nature16961

[5] Siri Team, "Hey Siri: An On-device DNN-powered Voice Trigger for Apple's Personal Assistant," *Apple Machine Learning Research*, 2017. [Online]. Available: https://machinelearning.apple.com/research/hey-siri

[6] T. B. Brown, B. Mann *et al.*, "Language models are few-shot learners," 2020. [Online]. Available: https://arxiv.org/abs/2005.14165

[7] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with clip latents," 2022. [Online]. Available: https://arxiv.org/abs/2204.06125

[8] Tesla, "Autopilot and Full Self-Driving Capability," 2022. [Online]. Available: https://www.tesla.com/support/autopilot

[9] G. Ras, N. Xie, M. van Gerven, and D. Doran, "Explainable deep learning: A field guide for the uninitiated," 2020. [Online]. Available: https://arxiv.org/abs/2004.14545

[10] K. Hao, "AI is sending people to jail—and getting it wrong," 2019. [Online]. Available: https://www.technologyreview.com/2019/01/21/137783/algorithms-criminal-justice-ai/

[11] R. Rao, "The Dutch Tax Authority Was Felled by AI—What Comes Next? European regulation hopes to rein in ill-behaving algorithms," 2022. [Online]. Available: https://spectrum.ieee.org/artificial-intelligence-in-government

[12] N. E. Boudette, "'It Happened So Fast': Inside a Fatal Tesla Autopilot Accident," 2021. [Online]. Available: https://www.nytimes.com/2021/08/17/business/tesla-autopilot-accident.html

[13] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, A. Singh and J. Zhu, Eds., vol. 54. PMLR, 20–22 Apr 2017, pp. 1273–1282. [Online]. Available: https://proceedings.mlr.press/v54/mcmahan17a.html

[14] Q. Yang, Y. Liu, Y. Cheng, Y. Kang, T. Chen, and H. Yu, "Federated learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 13, no. 3, pp. 1–207, 2019.

[15] researchandmarkets.com, "Global Federated Learning Market by Application (Drug Discovery, Industrial IoT, Risk Management), Vertical (Healthcare & Life Sciences, BFSI, Manufacturing, Automotive & Transportation, Energy & Utilities), and Region - Forecast to 2028," 2022. [Online]. Available: https://www.researchandmarkets.com/reports/5311706

[16] N. Rieke, J. Hancox *et al.*, "The future of digital health with federated learning," *npj Digital Medicine*, vol. 3, no. 1, p. 119, Sep 2020. [Online]. Available: https://doi.org/10.1038/s41746-020-00323-1

[17] V. Dignum, *Responsible artificial intelligence: how to develop and use AI in a responsible way.* Springer Nature, 2019.

[18] L. Floridi, J. Cowls *et al.*, "AI4People - An Ethical Framework for a Good AI Society: Opportunities, Risks, Principles, and Recommendations," *Minds and Machines*, vol. 28, no. 4, pp. 689–707, Dec 2018. [Online]. Available: https://doi.org/10.1007/s11023-018-9482-5

[19] S. Feuerriegel, M. Dolata, and G. Schwabe, "Fair AI," *Business & information systems engineering*, vol. 62, no. 4, pp. 379–384, 2020.

[20] AI HLEG of the European Commission, "Ethics guidelines for trustworthy ai," 2019. [Online]. Available: https://ec.europa.eu/futurium/en/ai-alliance-consultation.1.html

[21] H. Liu, Y. Wang *et al.*, "Trustworthy AI: A Computational Perspective," 2021. [Online]. Available: https://arxiv.org/abs/2107.06641

[22] B. Li, P. Qi *et al.*, "Trustworthy AI: From Principles to Practices," *ACM Comput. Surv.*, aug 2022, just Accepted. [Online]. Available: https://doi.org/10.1145/3555803

[23] S. Thiebes, S. Lins, and A. Sunyaev, "Trustworthy artificial intelligence," *Electronic Markets*, vol. 31, no. 2, pp. 447–464, 2021.

[24] D. Kaur, S. Uslu, K. J. Rittichier, and A. Durresi, "Trustworthy artificial intelligence: a review," *ACM Computing Surveys (CSUR)*, vol. 55, no. 2, pp. 1–38, 2022.

[25] K. R. Varshney, "Trustworthy machine learning and artificial intelligence," *XRDS: Crossroads, The ACM Magazine for Students*, vol. 25, no. 3, pp. 26–29, 2019.

[26] J. M. Wing, "Trustworthy AI," *Communications of the ACM*, vol. 64, no. 10, pp. 64–71, 2021.

[27] P. Kairouz, H. B. McMahan *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

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

[29] Adapt, "Flower: A Friendly Federated Learning Framework." [Online]. Available: https://flower.dev/

[30] D. Dimitriadis, M. Hipolito Garcia, D. Madrigal, A. Manoel, and R. Sim, "FLUTE: A Scalable, Extensible Framework for High-Performance Federated Learning Simulations," March 2022. [Online]. Available: https://www.microsoft.com/en-us/research/publication/flute-a-scalable-extensible-framework-for-high-performance-federated-learning-simulations/

[31] S. Caldas, S. M. K. Duddu *et al.*, "Leaf: A benchmark for federated settings," 2018. [Online]. Available: https://arxiv.org/abs/1812.01097

[32] Y. Xie, Z. Wang, D. Chen, D. Gao, L. Yao, W. Kuang, Y. Li, B. Ding, and J. Zhou, "Federatedscope: A flexible federated learning platform for heterogeneity," *arXiv preprint arxiv.2204.05011*, 2022.

[33] Alibaba DAMO Academy, "Data Analytics and Intelligence Lab." [Online]. Available: https://damo.alibaba.com/labs/data-analytics-and-intelligence

[34] Xie, Ning, "FederatedTrust: A Trustworthiness Evaluation Framework," 2022. [Online]. Available: https://github.com/ningxie1991/FederatedTrust

[35] N. Carlini, A. Athalye, N. Papernot, W. Brendel, J. Rauber, D. Tsipras, I. Goodfellow, A. Madry, and A. Kurakin, "On evaluating adversarial robustness," *arXiv preprint arXiv:1902.06705*, 2019.

[36] M. Al-Rubaie and J. M. Chang, "Privacy-preserving machine learning: Threats and solutions," *IEEE Security & Privacy*, vol. 17, no. 2, pp. 49–58, 2019.

[37] B. Liu, M. Ding, S. Shaham, W. Rahayu, F. Farokhi, and Z. Lin, "When machine learning meets privacy: A survey and outlook," *ACM Computing Surveys (CSUR)*, vol. 54, no. 2, pp. 1–36, 2021.

[38] I. Wagner and D. Eckhoff, "Technical privacy metrics: a systematic survey," *ACM Computing Surveys (CSUR)*, vol. 51, no. 3, pp. 1–38, 2018.

[39] L. Mehner, S. N. von Voigt, and F. Tschorsch, “Towards Explaining Epsilon: A Worst-Case Study of Differential Privacy Risks,” in *2021 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW).* IEEE, 2021, pp. 328–331.

[40] D. Pessach and E. Shmueli, “A review on fairness in machine learning,” *ACM Computing Surveys (CSUR)*, vol. 55, no. 3, pp. 1–44, 2022.

[41] F. Doshi-Velez and B. Kim, “Towards a rigorous science of interpretable machine learning,” *arXiv preprint arXiv:1702.08608*, 2017.

[42] A. B. Arrieta *et al.*, “Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai,” 2019. [Online]. Available: https://arxiv.org/abs/1910.10045

[43] M. T. Ribeiro, S. Singh, and C. Guestrin, "“Why should I trust you?” Explaining the predictions of any classifier,” in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[44] A. Blanco-Justicia, J. Domingo-Ferrer, S. Martinez, and D. Sanchez, “Machine learning explainability via microaggregation and shallow decision trees,” *Knowledge-Based Systems*, vol. 194, p. 105532, 2020.

[45] S. M. Lundberg and S.-I. Lee, “A unified approach to interpreting model predictions,” *Advances in neural information processing systems*, vol. 30, 2017.

[46] M. Wieringa, “What to account for when accounting for algorithms: a systematic literature review on algorithmic accountability,” in *Proceedings of the 2020 conference on fairness, accountability, and transparency*, 2020, pp. 1–18.

[47] I. D. Raji, A. Smart, R. N. White, M. Mitchell, T. Gebru, B. Hutchinson, J. Smith-Loud, D. Theron, and P. Barnes, “Closing the ai accountability gap: Defining an end-to-end framework for internal algorithmic auditing,” in *Proceedings of the 2020 conference on fairness, accountability, and transparency*, 2020, pp. 33–44.

[48] M. Arnold, R. K. Bellamy *et al.*, “Factsheets: Increasing trust in ai services through supplier’s declarations of conformity,” *IBM Journal of Research and Development*, vol. 63, no. 4/5, pp. 6–1, 2019.

[49] A. Celdran, J. Bauer, M. Demirci *et al.*, “Quantifying Trustworthiness of Supervised Machine and Deep Learning Models,” *UZH Master Project*, 2022.

[50] IBM Research, “The AI 360 Toolkit: AI models explained,” 2022. [Online]. Available: https://developer.ibm.com/articles/the-ai-360-toolkit-ai-models-explained/

[51] Q. Yang, Y. Liu, T. Chen, and Y. Tong, “Federated machine learning: Concept and applications,” *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[52] S. K. Lo, Q. Lu *et al.*, “Architectural patterns for the design of federated learning systems,” *Journal of Systems and Software*, vol. 191, p. 111357, 2022.

[53] X. Yin, Y. Zhu, and J. Hu, "A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions," *ACM Computing Surveys (CSUR)*, vol. 54, no. 6, pp. 1–36, 2021.

[54] Q. Li, Z. Wen, Z. Wu, S. Hu, N. Wang, Y. Li, X. Liu, and B. He, "A survey on federated learning systems: vision, hype and reality for data privacy and protection," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[55] A. Nilsson, S. Smith, G. Ulm, E. Gustavsson, and M. Jirstrand, "A performance evaluation of federated learning algorithms," in *Proceedings of the second workshop on distributed infrastructures for deep learning*, 2018, pp. 1–8.

[56] J. Konečnỳ, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *arXiv preprint arXiv:1610.02527*, 2016.

[57] Y. Wang, "Co-op: Cooperative machine learning from mobile devices," *University of Alberta Libraries*, 2017.

[58] S. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečnỳ, S. Kumar, and H. B. McMahan, "Adaptive federated optimization," *arXiv preprint arXiv:2003.00295*, 2020.

[59] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proceedings of Machine Learning and Systems*, vol. 2, pp. 429–450, 2020.

[60] X. Li, M. Jiang, X. Zhang, M. Kamp, and Q. Dou, "FedBN: Federated learning on non-iid features via local batch normalization," *arXiv preprint arXiv:2102.07623*, 2021.

[61] C. T Dinh, N. Tran, and J. Nguyen, "Personalized federated learning with moreau envelopes," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21 394–21 405, 2020.

[62] M. S. Jere, T. Farnan, and F. Koushanfar, "A taxonomy of attacks on federated learning," *IEEE Security & Privacy*, vol. 19, no. 2, pp. 20–28, 2020.

[63] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *International Conference on Machine Learning.* PMLR, 2019, pp. 634–643.

[64] M. Naseri, J. Hayes, and E. De Cristofaro, "Local and central differential privacy for robustness and privacy in federated learning," *arXiv preprint arXiv:2009.03561*, 2020.

[65] L. Muñoz-González, K. T. Co, and E. C. Lupu, "Byzantine-robust federated machine learning through adaptive model averaging," *arXiv preprint arXiv:1909.05125*, 2019.

[66] N. Rodríguez-Barroso, E. Martínez-Cámara, M. V. Luzón, and F. Herrera, "Backdoor attacks-resilient aggregation based on Robust Filtering of Outliers in federated learning for image classification," *Knowledge-Based Systems*, vol. 245, p. 108588, 2022.

[67] M. Alfarra, J. C. Pérez, E. Shulgin, P. Richtárik, and B. Ghanem, "Certified Robustness in Federated Learning," *arXiv preprint arXiv:2206.02535*, 2022.

[68] Y. Dong, X. Chen, L. Shen, and D. Wang, "EaSTFLy: Efficient and secure ternary federated learning," *Computers & Security*, vol. 94, p. 101824, 2020.

[69] S. Hardy, W. Henecka *et al.*, "Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption," *arXiv preprint arXiv:1711.10677*, 2017.

[70] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1175–1191.

[71] O. Choudhury, A. Gkoulalas-Divanis, T. Salonidis, I. Sylla, Y. Park, G. Hsu, and A. Das, "Differential privacy-enabled federated learning for sensitive health data," *arXiv preprint arXiv:1910.02578*, 2019.

[72] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," *arXiv preprint arXiv:1712.07557*, 2017.

[73] O. Choudhury, A. Gkoulalas-Divanis *et al.*, "A syntactic approach for privacy-preserving federated learning," in *ECAI 2020*. IOS Press, 2020, pp. 1762–1769.

[74] Y. Liu, X. Zhu, J. Wang, and J. Xiao, "A quantitative metric for privacy leakage in federated learning," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 3065–3069.

[75] Y. Shi, H. Yu, and C. Leung, "A survey of fairness-aware federated learning," *arXiv preprint arXiv:2111.01872*, 2021.

[76] X. Yue, M. Nouiehed, and R. A. Kontar, "Gifair-fl: An approach for group and individual fairness in federated learning," *arXiv preprint arXiv:2108.02741*, 2021.

[77] D. Y. Zhang, Z. Kou, and D. Wang, "Fairfl: A fair federated learning approach to reducing demographic bias in privacy-sensitive classification models," in *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, 2020, pp. 1051–1060.

[78] T. Huang, W. Lin, W. Wu, L. He, K. Li, and A. Y. Zomaya, "An efficiency-boosting client selection scheme for federated learning with fairness guarantee," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 7, pp. 1552–1564, 2020.

[79] Z. Fan, H. Fang, Z. Zhou, J. Pei, M. P. Friedlander, C. Liu, and Y. Zhang, "Improving fairness for data valuation in horizontal federated learning," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2022, pp. 2440–2453.

[80] G. Wang, “Interpret federated learning with shapley values,” *arXiv preprint arXiv:1905.04519*, 2019.

[81] C. Molnar, *Interpretable machine learning.* BOOKDOWN, 2020. [Online]. Available: https://christophm.github.io/interpretable-ml-book/shapley.html

[82] P. Chen, X. Du, Z. Lu, J. Wu, and P. C. Hung, “Evfl: An explainable vertical federated learning for data-oriented artificial intelligence systems,” *Journal of Systems Architecture*, vol. 126, p. 102474, 2022.

[83] R. Guidotti, “Counterfactual explanations and how to find them: literature review and benchmarking,” *Data Mining and Knowledge Discovery*, pp. 1–55, 2022.

[84] T. van Erven and P. Harremos, “Rényi divergence and kullback-leibler divergence,” *IEEE Transactions on Information Theory*, vol. 60, no. 7, pp. 3797–3820, 2014.

[85] N. Baracaldo, A. Anwar, M. Purcell, A. Rawat, M. Sinn, B. Altakrouri, D. Balta, M. Sellami, P. Kuhn, U. Schopp *et al.*, “Towards an accountable and reproducible federated learning: A factsheets approach,” *arXiv preprint arXiv:2202.12443*, 2022.

[86] H. B. Desai, M. S. Ozdayi, and M. Kantarcioglu, “Blockfla: Accountable federated learning via hybrid blockchain architecture,” in *Proceedings of the eleventh ACM conference on data and application security and privacy*, 2021, pp. 101–112.

[87] V. Mugunthan, R. Rahman, and L. Kagal, “Blockflow: An accountable and privacy-preserving solution for federated learning,” *arXiv preprint arXiv:2007.03856*, 2020.

[88] S. Awan, F. Li, B. Luo, and M. Liu, “Poster: A reliable and accountable privacy-preserving federated learning framework using the blockchain,” in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 2561–2563.

[89] D. Chai, L. Wang, K. Chen, and Q. Yang, “Fedeval: A benchmark system with a comprehensive evaluation model for federated learning,” *arXiv preprint arXiv:2011.09655*, 2020.

[90] IBM Research, “Ai factsheets 360.” [Online]. Available: https://aifs360.mybluemix.net/

[91] L. Lyu, H. Yu, X. Ma, L. Sun, J. Zhao, Q. Yang, and P. S. Yu, “Privacy and robustness in federated learning: Attacks and defenses,” *arXiv preprint arXiv:2012.06337*, 2020.

[92] J. Cohen, E. Rosenfeld, and Z. Kolter, “Certified adversarial robustness via randomized smoothing,” in *International Conference on Machine Learning.* PMLR, 2019, pp. 1310–1320.

[93] C. Wu, X. Yang, S. Zhu, and P. Mitra, “Mitigating backdoor attacks in federated learning,” *arXiv preprint arXiv:2011.01767*, 2020.

[94] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *International Conference on Artificial Intelligence and Statistics.* PMLR, 2020, pp. 2938–2948.

[95] T.-W. Weng, H. Zhang, P.-Y. Chen, J. Yi, D. Su, Y. Gao, C.-J. Hsieh, and L. Daniel, "Evaluating the robustness of neural networks: An extreme value theory approach," *arXiv preprint arXiv:1801.10578*, 2018.

[96] J. Liang, M. Wang, Y. Chen, Y. Jiang, and R. Zhang, "Fuzz testing in practice: Obstacles and solutions," in *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 2018, pp. 562–566.

[97] S. S. Malladi and H. C. Subramanian, "Bug bounty programs for cybersecurity: Practices, issues, and recommendations," *IEEE Software*, vol. 37, no. 1, pp. 31–39, 2019.

[98] S. U. Farooq, S. Quadri, and N. Ahmad, "Metrics, models and measurements in software reliability," in *2012 IEEE 10th international symposium on applied machine intelligence and informatics (SAMI).* IEEE, 2012, pp. 441–449.

[99] H. Zhu, J. Xu, S. Liu, and Y. Jin, "Federated learning on non-IID data: A survey," *Neurocomputing*, vol. 465, pp. 371–390, 2021.

[100] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," *Advances in neural information processing systems*, vol. 30, 2017.

[101] F. Sattler, K.-R. Müller, and W. Samek, "Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 8, pp. 3710–3722, 2020.

[102] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, and S. Choudhary, "Federated learning with personalization layers," *arXiv preprint arXiv:1912.00818*, 2019.

[103] J. Kang, Z. Xiong, D. Niyato, Y. Zou, Y. Zhang, and M. Guizani, "Reliable federated learning for mobile networks," *IEEE Wireless Communications*, vol. 27, no. 2, pp. 72–80, 2020.

[104] B. Pejó and G. Biczók, "Quality inference in federated learning with secure aggregation," *arXiv preprint arXiv:2007.06236*, 2020.

[105] S. Divi, Y.-S. Lin, H. Farrukh, and Z. B. Celik, "New metrics to evaluate the performance and fairness of personalized federated learning," *arXiv preprint arXiv:2107.13173*, 2021.

[106] X. Yin, Y. Zhu, and J. Hu, "A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions," *ACM Computing Surveys (CSUR)*, vol. 54, no. 6, pp. 1–36, 2021.

[107] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. Quek, and H. V. Poor, "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3454–3469, 2020.

[108] T. Li, M. Sanjabi, A. Beirami, and V. Smith, "Fair resource allocation in federated learning," *arXiv preprint arXiv:1905.10497*, 2019.

[109] M. Yang, X. Wang, H. Zhu, H. Wang, and H. Qian, "Federated learning with class imbalance reduction," in *2021 29th European Signal Processing Conference (EUSIPCO)*. IEEE, 2021, pp. 2174–2178.

[110] S. R. Islam, W. Eberle, and S. K. Ghafoor, "Towards quantification of explainability in explainable artificial intelligence methods," in *The thirty-third international flairs conference*, 2020.

[111] R. Haffar, D. Sánchez, and J. Domingo-Ferrer, "Explaining predictions and attacks in federated learning via random forests," *Applied Intelligence*, pp. 1–17, 2022.

[112] P. Cortez and M. J. Embrechts, "Opening black box data mining models using sensitivity analysis," in *2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, 2011, pp. 341–348.

[113] M. Ungersböck, T. Hiessl *et al.*, "Explainable Federated Learning: A Lifecycle Dashboard for Industrial Settings," *TechRxiv*, 2022.

[114] N. Kohli, R. Barreto, and J. A. Kroll, "Translation tutorial: a shared lexicon for research and practice in human-centered software systems," in *1st Conference on Fairness, Accountability, and Transparancy. New York, NY, USA*, vol. 7, 2018.

[115] IBM AI FactSheets 360, "Accountable Federated Learning: A Classifying Citizen Participation Ideas Use Case," 2022. [Online]. Available: https://aifs360.mybluemix.net/examples/federated_learning

[116] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[117] A. Ziller, A. Trask, A. Lopardo, B. Szymkow, B. Wagner, E. Bluemke, J.-M. Nounahon, J. Passerat-Palmbach, K. Prakash, N. Rose, T. Ryffel, Z. N. Reza, and G. Kaissis, *PySyft: A Library for Easy Federated Learning.* Cham: Springer International Publishing, 2021, pp. 111–139. [Online]. Available: https://doi.org/10.1007/978-3-030-70604-3_5

[118] OpenMinded, "Federated Learning." [Online]. Available: https://blog.openmined.org/tag/federated-learning/

[119] C. He, S. Li, J. So, X. Zeng, M. Zhang, H. Wang, X. Wang, P. Vepakomma, A. Singh, H. Qiu *et al.*, "Fedml: A research library and benchmark for federated machine learning," *arXiv preprint arXiv:2007.13518*, 2020.

Abbreviations

AI	Artificial Intelligence
ART	Adversarial Robustness 360 Toolbox
CCPA	California Consumer Privacy Act
CNN	Convolutional Neural Network
CV	Coefficient of Variation
DAIL	Data Analytics and Intelligence Lab
DL	Deep Learning
DNN	Deep Neural Network
DP	Differential Privacy
EU	European Union
FedAvg	FederatedAveraging
FL	Federated Learning
FTL	Federated Transfer Learning
GAN	Generative Adversarial Networks
GPT-3	Generative Pre-trained Transformer 3
HFL	Horizontal Federated Learning
IBM	International Business Machines Corporation
KL	Kullback-Leibler
MARL	Multi-agent Reinforcement Learning
ML	Machine Learning
ML/DL	Machine and Deep Learning
NbAFL	Noising before model Aggregation FL

PPFL Privacy-preserving Federated Learning

RAI Responsible AI

Siri Speech Interpretation and Recognition Interface

SMC Secure Multiparty Computation

SVM Support Vector Machine

TFF TensorFlow Federated

VFL Vertical Federated Learning

XAI explainable AI

Glossary

AI HLEG High-Level Expert Group on Artificial Intelligence

Big Data Big data refers to vast amounts of data that traditional storage methods cannot handle.

DALL·E 2 DALL. E 2 is a new AI system that can create realistic images and art from a description in natural language

List of Figures

3.1 Trustworthy FL Taxonomy 37

4.1 Overview of an FL round implemented with FederatedScope 41

4.2 Overview of the interaction between FederatedScope with FederatedTrust . 43

4.3 Reduced Taxonomy for Prototype v1 46

4.4 Metric Object Structure 53

4.5 Code Structure and UML Class Diagram 62

4.6 Design for the FederatedTrust Algorithm 63

A.1 Transparent ML Models 88

A.2 Feature Importance Calculation for HFL 89

A.3 Feature Importance Calculation for VFL 89

A.4 Shapley Value Visualization 90

A.5 Feature Importance Visualization on Images 90

A.6 Information Leakage Risk H-MINE Algorithm 91

List of Tables

4.1 Metrics for Robustness 48

4.2 Metrics for Privacy 48

4.3 Metrics for Fairness 48

4.4 Metrics for Explainability 49

4.5 Metrics for Accountability 49

4.6 Metrics for Architectural Soundness 49

4.7 Use case Basic Configurations 65

4.8 Experiment 1 Setting 65

4.9 Experiment 2 Setting 65

4.10 Experiment 3 Setting 66

4.11 Experiment 4 Setting 66

4.12 Common Results in the Experiments 67

4.13 Experiment 1 - 10 clients, 20% sampling rate, 5 rounds, no DP 68

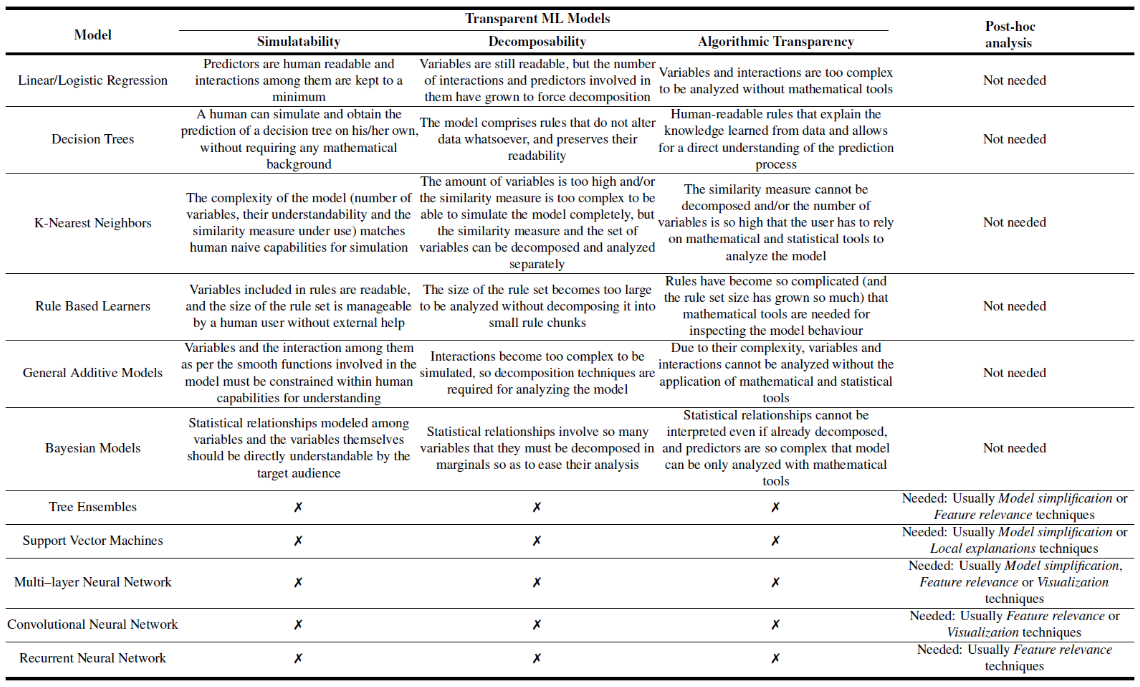
4.14 Experiment 2 - 50 clients, 60% sampling rate, 25 rounds, DP with $\epsilon = 20$. 68

4.15 Experiment 3 - 50 clients, 60% sampling rate, 20 rounds, DP with $\epsilon = 6$ . . 70

4.16 Experiment 4 - 100 clients, 60% sampling rate, 50 rounds, DP with $\epsilon = 6$. 70

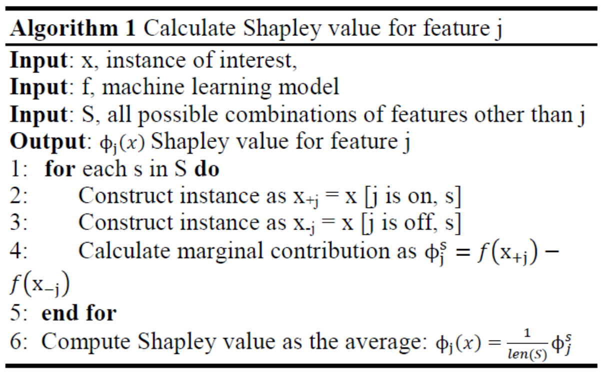
Appendix A

Metric Calculation References



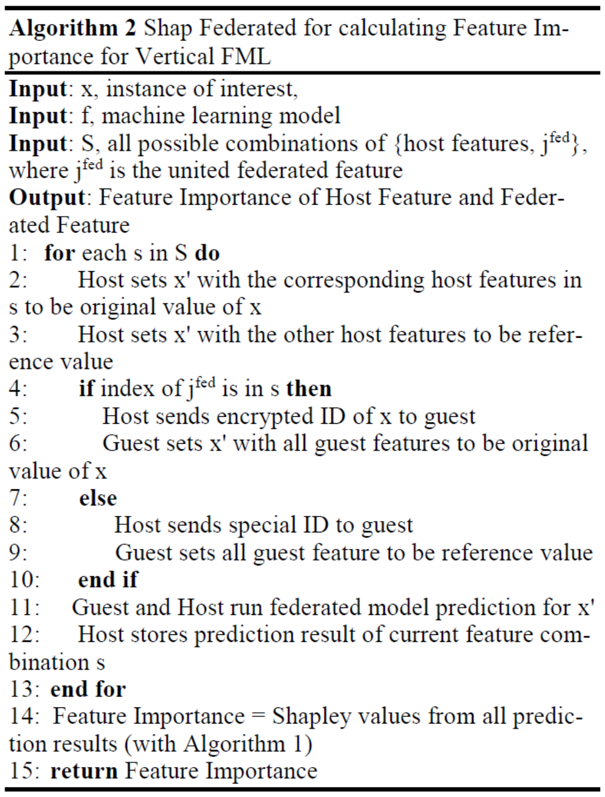
Model	Transparent ML Models			Post-hoc analysis
	Simulatability	**Decomposability**	**Algorithmic Transparency**	
Linear/Logistic Regression	Predictors are human readable and interactions among them are kept to a minimum	Variables are still readable, but the number of interactions and predictors involved in them have grown to force decomposition	Variables and interactions are too complex to be analyzed without mathematical tools	Not needed
Decision Trees	A human can simulate and obtain the prediction of a decision tree on his/her own, without requiring any mathematical background	The model comprises rules that do not alter data whatsoever, and preserves their readability	Human-readable rules that explain the knowledge learned from data and allows for a direct understanding of the prediction process	Not needed
K-Nearest Neighbors	The complexity of the model (number of variables, their understandability and the similarity measure under use) matches human naive capabilities for simulation	The amount of variables is too high and/or the similarity measure is too complex to be able to simulate the model completely, but the similarity measure and the set of variables can be decomposed and analyzed separately	The similarity measure cannot be decomposed and/or the number of variables is so high that the user has to rely on mathematical and statistical tools to analyze the model	Not needed
Rule Based Learners	Variables included in rules are readable, and the size of the rule set is manageable by a human user without external help	The size of the rule set becomes too large to be analyzed without decomposing it into small rule chunks	Rules have become so complicated (and the rule set size has grown so much) that mathematical tools are needed for inspecting the model behaviour	Not needed
General Additive Models	Variables and the interaction among them as per the smooth functions involved in the model must be constrained within human capabilities for understanding	Interactions become too complex to be simulated, so decomposition techniques are required for analyzing the model	Due to their complexity, variables and interactions cannot be analyzed without the application of mathematical and statistical tools	Not needed
Bayesian Models	Statistical relationships modeled among variables and the variables themselves should be directly understandable by the target audience	Statistical relationships involve so many variables that they must be decomposed in marginals so as to ease their analysis	Statistical relationships cannot be interpreted even if already decomposed, and predictors are so complex that model can be only analyzed with mathematical tools	Not needed
Tree Ensembles	✗	✗	✗	Needed: Usually *Model simplification* or *Feature relevance* techniques
Support Vector Machines	✗	✗	✗	Needed: Usually *Model simplification* or *Local explanations* techniques
Multi–layer Neural Network	✗	✗	✗	Needed: Usually *Model simplification*, *Feature relevance* or *Visualization* techniques
Convolutional Neural Network	✗	✗	✗	Needed: Usually *Feature relevance* or *Visualization* techniques
Recurrent Neural Network	✗	✗	✗	Needed: Usually *Feature relevance* techniques

Figure A.1: Transparent ML Models



```
Algorithm 1 Calculate Shapley value for feature j
Input: x, instance of interest,
Input: f, machine learning model
Input: S, all possible combinations of features other than j
Output: $\phi_j(x)$ Shapley value for feature j
1: for each s in S do
2:     Construct instance as $x_{+j}$ = x [j is on, s]
3:     Construct instance as $x_{-j}$ = x [j is off, s]
4:     Calculate marginal contribution as $\phi_j^s = f(x_{+j}) - f(x_{-j})$
5: end for
6: Compute Shapley value as the average: $\phi_j(x) = \frac{1}{len(S)} \phi_j^s$
```

Figure A.2: Feature Importance Calculation for HFL



```
Algorithm 2 Shap Federated for calculating Feature Importance for Vertical FML
Input: x, instance of interest,
Input: f, machine learning model
Input: S, all possible combinations of {host features, $j^{fed}$}, where $j^{fed}$ is the united federated feature
Output: Feature Importance of Host Feature and Federated Feature
1: for each s in S do
2:     Host sets x' with the corresponding host features in s to be original value of x
3:     Host sets x' with the other host features to be reference value
4:     if index of $j^{fed}$ is in s then
5:         Host sends encrypted ID of x to guest
6:         Guest sets x' with all guest features to be original value of x
7:     else
8:         Host sends special ID to guest
9:         Guest sets all guest feature to be reference value
10:    end if
11:    Guest and Host run federated model prediction for x'
12:    Host stores prediction result of current feature combination s
13: end for
14: Feature Importance = Shapley values from all prediction results (with Algorithm 1)
15: return Feature Importance
```

Figure A.3: Feature Importance Calculation for VFL

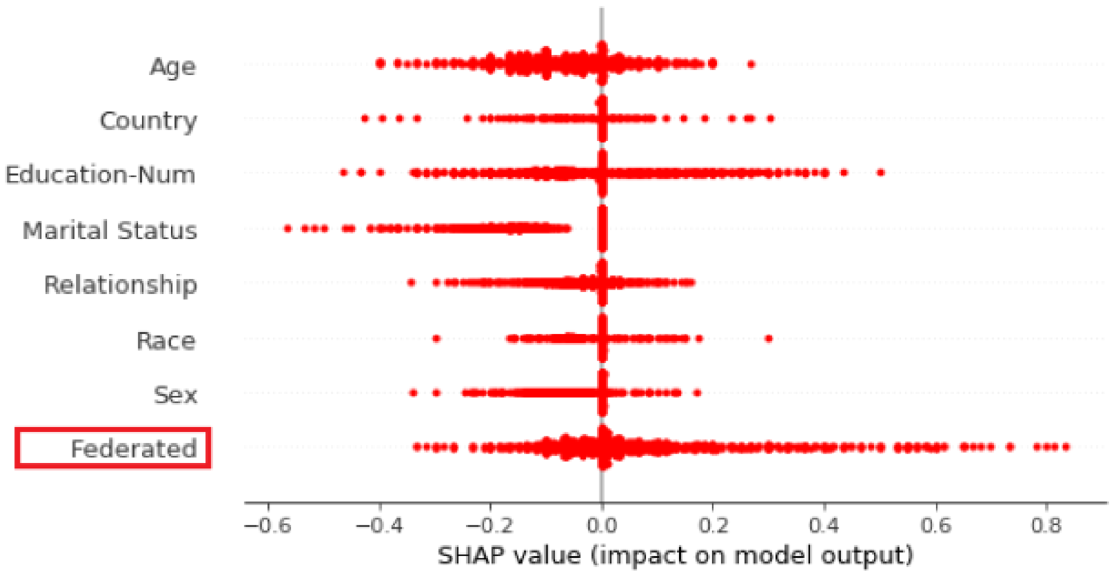


Figure A.4: Shapley Value Visualization

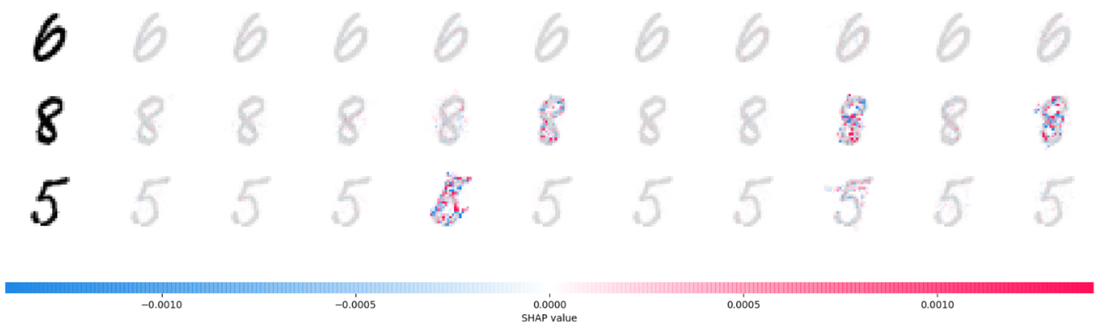


Figure A.5: Feature Importance Visualization on Images

```
Algorithm 1: H-MINE Algorithm.
Inputs: Task model batch size $B$, sample size $S$, task
        model status $\theta^t$.
Initialize: H-MINE parameters $\phi$.
1  while not converged do
2      for $k = 1, ..., S$ do
3          Generate batch samples $X_B, \hat{X}_B$:
4          $X_B \leftarrow [x_1, ..., x_B]$, $\hat{X}_B \leftarrow [\hat{x}_1, ..., \hat{x}_B]$
5          Calculate corresponding gradients:
6          $G = \nabla_\theta f_{\theta^t}(X_B)$, $\hat{G} = \nabla_\theta f_{\theta^t}(\hat{X}_B)$
7          for $j = 1, ..., B$ do
8              $h_j = \textbf{BlockModel}_\phi(x_j, G)$
9              $\hat{h}_j = \textbf{BlockModel}_\phi(x_j, \hat{G})$
10         end
11         $H = \{h_1, h_2, ..., h_B\}$, $\hat{H} = \{\hat{h}_1, \hat{h}_2, ..., \hat{h}_B\}$
12         $v_k = T_\phi(X_B, G) = \textbf{MixModel}_\phi(H, G)$,
13         $\hat{v}_k = T_\phi(X_B, \hat{G}) = \textbf{MixModel}_\phi(\hat{H}, \hat{G})$
14     end
15     Evaluate the lower-bound:
16     $V(\phi) = \frac{1}{S}\sum_{k=1}^{S} v_k - \log(\frac{1}{S}\sum_{k=1}^{S} e^{\hat{v}_k})$
17     Update H-MINE parameters:
18     $\phi \leftarrow \phi + \nabla_\phi V(\phi)$ ;
19 end
```

Figure A.6: Information Leakage Risk H-MINE Algorithm