University of Zurich^UZH

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

Designing and Implementing an Advanced Algorithm to Measure the Trustworthiness Level of Federated Learning Models

Lynn Zumtaugwald
Zurich
17-929-340

Supervisors: Dr. Alberto Huertas
Date of Submission: August 7, 2023

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

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

Artificial intelligence (AI) has immersed our daily lives and assists in the decision process of critical sectors such as medicine and law. Therefore it is now more important than ever before that AI systems developed are reliable, ethical, and do not cause harm to humans. The High-Level Expert Group on AI (AI-HLEG) of the European Commission has laid the foundation by defining seven key requirements for trustworthy AI systems. To address concerns about privacy risks associated with centralized learning approaches federated learning (FL) has emerged as a promising and widely used alternative. FL allows multiple clients to collaboratively train machine learning models without the need for sharing private data. Because of the high adaption of FL systems, ensuring that they are trustworthy is crucial. Previous research efforts have proposed a trustworthy FL taxonomy with six pillars, each comprehensively defined with notions and metrics. This taxonomy covers six of the seven requirements defined by the AI-HLEG. However, one notable aspect that has been largely overlooked by research is the requirement for environmental well-being in trustworthy AI/FL. This leaves a significant gap between the expectations set by governing bodies and the guidelines applied and measured by researchers. This master thesis addresses this gap by introducing the sustainability pillar to the trustworthy FL taxonomy and thus presenting the first taxonomy that comprehensively addresses all the requirements defined by the AI-HLEG. The sustainability pillar focuses on assessing the environmental impact of FL systems and incorporates three main aspects: hardware efficiency, federation complexity, and the carbon intensity of the energy grid, each with well-defined metrics. As a second contribution, this master thesis extends an existing prototype to evaluate the trustworthiness of FL systems with the sustainability pillar. The prototype is then extensively evaluated in various scenarios, involving different federation configurations. The results shed light on the trustworthiness of different federation configurations in different settings with varying complexities, hardware, and energy grids used. Importantly, the sustainability pillar's score corrects the overall trust score by considering the environmental impact of FL systems across seven key pillars. Thus, the proposed taxonomy and prototype are the first to comprehensively address all seven AI-HLEG requirements and lay the foundation for a more accurate trustworthiness assessment of FL systems.

Abstrakt

Künstliche Intelligenz (KI) hat unser tägliches Leben durchdrungen und unterstützt den Entscheidungsprozess in kritischen Bereichen wie Medizin und Recht. Daher ist es heute wichtiger denn je, dass die entwickelten KI-Systeme zuverlässig und ethisch vertretbar sind und dem Menschen keinen Schaden zufügen. Die High-Level Expert Group on AI (AI-HLEG) der Europäischen Kommission hat mit der Definition von sieben Schlüsselanforderungen für vertrauenswürdige KI-Systeme den Grundstein dafür gelegt. Um die Bedenken hinsichtlich der mit zentralisierten Lernansätzen verbundenen Risiken für den Datenschutz auszuräumen, hat sich das Federated Learning (FL) als vielversprechende und weit verbreitete Alternative erwiesen. FL ermöglicht es mehreren Teilnehmern, gemeinsam Modelle für maschinelles Lernen zu trainieren, ohne dass private Daten geteilt werden müssen. Aufgrund des weitverbreiteten Gebrauchs von FL-Systemen ist die Sicherstellung ihrer Vertrauenswürdigkeit von entscheidender Bedeutung. Frühere Forschungsarbeiten haben eine vertrauenswürdige FL-Taxonomie mit sechs Säulen vorgeschlagen, die jeweils umfassend mit Begriffen und Metriken definiert sind. Diese Taxonomie deckt sechs der sieben von der AI-HLEG definierten Anforderungen ab. Ein wichtiger Aspekt, der von der Forschung weitgehend übersehen wurde, ist jedoch die Anforderung an Nachhaltigkeit in vertrauenswürdiger KI/FL. Dies hinterlässt eine erhebliche Lücke zwischen den Erwartungen der Behörden und den von den Forschern angewandten und gemessenen Richtlinien. Die vorliegende Masterarbeit schließt diese Lücke, indem sie die Säule der Nachhaltigkeit in die Taxonomie vertrauenswürdiger KI/FL einführt und damit die erste Taxonomie vorstellt, die alle von der AI-HLEG definierten Anforderungen umfassend berücksichtigt. Die Nachhaltigkeitssäule konzentriert sich auf die Bewertung der Umweltauswirkungen von FL-Systemen und umfasst drei Hauptaspekte: Hardware-Effizienz, Komplexität und die Kohlenstoffintensität des Energienetzes, jeweils mit genau definierten Metriken. Als zweiten Beitrag erweitert diese Masterarbeit einen bestehenden Prototyp, um die Vertrauenswürdigkeit von FL-Systemen mit dem Nachhaltigkeitsaspekt zu bewerten. Der Prototyp wird in verschiedenen Szenarien mit unterschiedlichen Konfigurationen von FL Systemen ausgiebig evaluiert. Die Ergebnisse geben Aufschluss über die Vertrauenswürdigkeit verschiedener FL Systeme in verschiedenen Umgebungen mit unterschiedlicher Komplexität, Hardware und verwendeten Energienetzen. Wichtig ist, dass die Bewertung der Nachhaltigkeitssäule die Gesamtvertrauensbewertung korrigiert, indem die Umweltauswirkungen von FL-Systemen über sieben Schlüsselsäulen hinweg berücksichtigt werden. Die vorgeschlagene Taxonomie und der Prototyp sind somit die ersten, die alle sieben AI-HLEG-Anforderungen umfassend berücksichtigen und den Grundstein für eine genauere Bewertung der Vertrauenswürdigkeit von FL-Systemen legen.

Acknowledgments

I would like to thank the Communication Systems Group at the University of Zurich for providing me the opportunity to conduct this insightful and exciting master's thesis in their research department and to contribute to the research community of trustworthy federated learning.

Further, I want to extend special thanks to Dr. Alberto Huertas, Chao Feng, and Prof. Dr. Stiller for their ongoing mentorship, guidance, and feedback during the research, implementation, and writing process of this master thesis. I am deeply grateful for their time and effort in helping me to complete this thesis.

Lastly, since this thesis concludes my studies at university, I would like to thank my family and friends. Their belief in me kept my spirits and motivation high and I am sincerely grateful for their love and support during this process and throughout my life.

Contents

Chapter 1

Introduction

1.1 Motivation

In the last decade, Artificial Intelligence (AI) has become omnipresent in humanity's daily lives. From playing Go against AlphaGo [1], using voice assistants like Apple's Siri [2] or Amazon's Alexa [2] to turn on the lights or play music, enjoying movies and series that fit you better using Netflix's recommender system [3] to getting ideas from ChatGPT [4] helping you to write essays or even write your programming code. AI's integration into critical domains like medicine, HR, and law has opened opportunities to predict cancer risks [5], assess job applicants [6], and even predict the risk of future crimes of a human [7].

However, with all the benefits AI has brought forth, it also introduced potential risks of danger. Different studies have shown racial bias in AI models that were used to predict if a human is in need and entitled to receive special medical care [8], refusing people of color the medical care they would have needed. Moreover, AI algorithms used to predict if a defendant would become a recidivist [9], falsely predicted people of color as high risk twice as often as their white counterparts. Gender bias was found in AI algorithms that were widely in use to predict a job appliance fit for the job [10], predicting women as less suitable for jobs in which men are more prevalent. Even if all the bias in the aforementioned AI systems has been mitigated, more examples of biased, harmful, or unsafe AI are found again and again. For example, Tesla's autopilot AI has been involved in 736 crashes and 17 fatalities since the year 2019 [11]. This shows the necessity of developing AI in a safe and trustworthy way to prevent unfairness, danger, and harm to humanity introduced by AI systems in the first place.

To ensure that AI is developed such that it is trustworthy and does not harm humans in any way, a new research field called trustworthy AI has evolved. Governing bodies such as the High-level Expert Group on Artificial Intelligence (AI-HLEG) [12] in Europe have established laws and guidelines to define trustworthy AI and build trustworthy AI. The seven requirements for trustworthy AI defined by the AI-HLEG [12] are human agency and oversight, technical robustness and safety, privacy and data governance, transparency, fairness, environmental well-being, and accountability.

In parallel, another crucial area of concern has caught attention - data privacy. Safeguarding human and organizational private data has become increasingly important. However, traditional centralized machine learning requires all the data to train a machine learning (ML) model in one single place, posing privacy risks. To mitigate that risk, Google introduced Federated Learning (FL) in 2016 [13], a novel approach where multiple clients collaboratively train an ML model without the need of sharing their private data by only sharing and aggregating model parameters.

Despite FL being privacy-preserving to a certain degree by design, trustworthiness remains a crucial factor in FL systems. Early works [14] in this field have laid the foundation by defining a trustworthy FL taxonomy and implementing a prototype to evaluate the trustworthiness of FL systems including the six pillars privacy, robustness, fairness, accountability, explainability, and federation. These pillars cover six of the seven requirements defined by the AI-HLEG. However, comparisons between the requirements defined by the AI-HLEG and the guidelines defined and the taxonomies show that one important aspect of trustworthy FL has been overlooked: the requirement of environmental well-being defined by the AI-HLEG. The AI-HLEG states in this requirement: ”AI systems should benefit all human beings, including future generations. It must hence be ensured that they are sustainable and environmentally friendly. Moreover, they should take into account the environment, including other living beings, and their social and societal impact should be carefully considered. [12, p. 19]”. Since the environmental impact of FL systems has not yet been considered in any trustworthy FL taxonomy, this leaves a significant gap between the expectations set by governing bodies and the guidelines applied and measured by researchers.

In order to close this gap and to align the trustworthy FL taxonomies and tools to the regulations defined, the environmental impact of FL systems needs to be included. To achieve this, a new pillar with appropriate metrics and notions needs to be defined, and a tool needs to be developed that evaluates and grades the environmental impact of an FL system as part of its trustworthiness.

1.2 Description of Work

Thus, the main contributions of this thesis are surveying the state-of-the-art literature to identify the missing aspect of environmental well-being in current trustworthy FL taxonomies, extending the taxonomy with a sustainability pillar representing the environmental impact of FL systems, defining metrics and notions that represent the environmental impact of FL systems and implementing the sustainability pillar an algorithmic prototype, resulting in the first trustworthy FL taxonomy and the first algorithmic prototype comprehensively assessing all the seven requirements for trustworthy AI defined by the AI-HLEG. In more detail, this master's thesis focuses on:

1. Literature Survey and Identification of missing aspects:
    - State-of-the-art literature to trustworthy FL is surveyed including the pillars: Privacy, Robustness, Fairness, Accountability, Explainability, and Federation
    - Laws and regulations to trustworthy AI/FL are revised
    - Existing trustworthy FL taxonomies from literature is analyzed and compared with the literature survey as well as the regulations and guidelines defined by governing bodies to identify missing aspects
    - Existing trustworthy FL evaluation tools are analyzed and compared with the literature survey as well as the regulations and guidelines defined by governing bodies to identify missing aspects
    - Research done to estimate the environmental impact of AI/FL system is reviewed
2. Establishment of a trustworthy FL taxonomy
    - The missing aspect of environmental impact is included into the trustworthy FL taxonomy by a new pillar sustainability
    - Notions and metrics representing the sustainability pillar are defined
    - The Limitations of the definition of the Sustainability Pillar are discussed
3. Design of an algorithmic prototype to evaluate the trustworthiness of FL models
    - The architecture of the algorithmic prototype is designed
    - Metrics and metric normalizations are designed
    - The trust score computation process is designed
    - The limitations of the design are discussed
4. Implementation of an algorithmic prototype to evaluate the trustworthiness of FL models:
    - The existing and most advanced prototype that evaluated the trustworthiness of FL systems in terms of privacy, robustness, fairness, accountability, explainability, and federation is extended with the new pillar of sustainability

- The package CodeCarbon is used to obtain information relevant to calculating metrics for the sustainability pillar
- CPU and GPU benchmarking datasets from PassMark are included in the algorithmic prototype to obtain the power performance of the hardware used by participants of the federation
- Limitations of the implementation are discussed

5. Evaluation of the algorithmic prototype

    - The algorithmic prototype is evaluated in parametrized and non-parametrized scenarios, showing that the prototype is able to correctly obtain the metrics for the sustainability pillar directly from the federation and its participants
    - Four different parametrized evaluation scenarios are run to evaluate the prototype's sustainability pillar behavior in situations where efficient or not efficient hardware, simple or complex federations, and not carbon-intensive or carbon-intensive energy grids are in place
    - Two evaluation scenarios are run where the algorithmic prototype is tested as a whole including all the seven pillars: privacy, robustness, fairness, explainability, accountability, federation, and sustainability showing the trustworthiness of different federations with varying configurations
    - The results of the experiments and the limitations are extensively discussed
    - Possible directions and ideas for future work are presented

1.3 Thesis Outline

Chapter 2 of this master thesis describes the context of FL and its classification as well as trustworthy AI, its pillars and laws, and guidelines defined. *Chapter 3* explores related work on trustworthy FL, existing trustworthy FL taxonomies, existing trustworthy FL evaluation tools, and related work on evaluating the environmental impact of AI/FL systems and identifies limitations. *Chapter 4* introduces the sustainability pillar to trustworthy FL including according notions and metrics. *Chapter 5* focuses on the design and implementation of an algorithmic prototype to evaluate the trustworthiness of FL systems. *Chapter 6* presents evaluation scenarios and their results. The end of the chapter contains a comparative discussion of all evaluation scenarios, their results, and the limitations of the current design and implementation. *Chapter 7* summarizes this master thesis, draws conclusions, and provides insights about future work. In the *Appendix A*, the algorithmic pseudocode for a previous algorithm that evaluated the trustworthiness of FL systems and that has been extended in this thesis is presented as well as information on the metrics that have already been defined.

Chapter 2

Background

This Chapter explains the basic technologies of FL (2.1) and its classification into horizontal (2.1.1), vertical (2.1.2), centralized (2.1.3) and decentralized FL (2.1.4). It continues with the need for trustworthy AI (2.2.1) and its five pillars (2.2.3) and discussed the requirements for trustworthy AI defined by the European Commission (2.2.2).

2.1 Federated Learning

Many of the past AI applications have been trained on a centralized approach where the data is stored in one place forcing users to share their data with a central entity [15]. Nowadays, data privacy is demanding an approach where users do not have to expose their data to a centralized entity in order to train an ML model. That's where FL comes into play. In the FL approach, ML models are trained over ten to potentially millions of distributed edge devices while keeping the client's data localized and preserving privacy through only sharing model parameters with the global model instead of the device's local data. This data privacy approach allows users to collaboratively train a model and benefit others without sharing their private data. The learning process of FL was categorized by Yang et al. [16] into Horizontal Federated Learning (HFL) and Vertical Federated Learning (VFL). Further, FL can be categorized into centralized federated learning (CFL) and decentralized federated learning (DFL) [17]. The following subsections elaborate on the different FL types.

2.1.1 Horizontal Federated Learning

In HFL, also referred to as homogenous FL, the datasets share the same feature space but differ in samples [18]. HFL, being the more commonly used type of FL, is applicable in scenarios where different devices or clients share the same consistent set of features. One well-known example is Google's KeyBoard prediction model, which is trained in an HFL setting with data coming from millions of devices [16].

2.1.2 Vertical Federated Learning

In VFL, also referred to as heterogenous FL, datasets share the sample ID space but differ in feature space [16]. Also for VFL, the goal is to collaboratively train a model, but using distributed data with different feature spaces. For example, a model to evaluate your overall health may be trained on data coming from the doctor, the hospital, your wearable fitness watch, and your mobile phone. All these data holders have different features of your health and they may be combined in a model using VFL [19].

2.1.3 Centralized Federated Learning

Centralized federated learning (CFL) is nowadays the predominant FL approach [17]. It consists of a central server to create and distribute the global models to the clients. The clients train the model on their local data and share the updated model parameters with the central server. The server updates the global model by aggregating all the local model parameters from the clients and distributing the global model again to the clients [20]. Figure 2.1 shows the five steps involved in a classical training cycle in CFL. In step one, a global model that is either pre-trained or initialized with weights at random is chosen on the central server. Then, in the second step, the initial model is distributed to the clients. Next, each client keeps training the model locally using its own local data. At some point, the weights of the locally trained and updated models are sent back from the clients to the central server, where the central server averages and aggregated them into a single, updated global model. As a last step, the updated global model is again distributed to all clients [21].

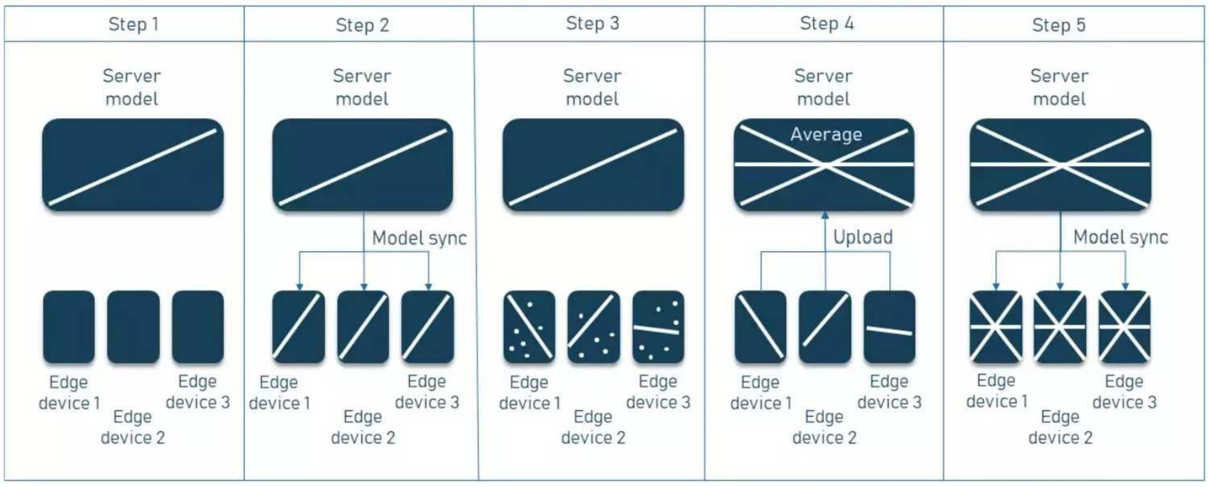


Figure 2.1: Federated Machine Learning in Steps from [21].

2.1.4 Decentralized Federated Learning

The newer approach of decentralized federated learning (DFL) [17], also defined as Decentralized FL, Distributed or Serverless FL removes the need for a centralized server by aggregation of model parameters from neighboring participants or nodes and emerged in 2018 [22]. Centralized approaches have the drawback that the central server can become a bottleneck and a single point of failure. DFL not using a centralized server removes this issue using decentralized model aggregation [23].

2.2 Trustworthy AI

2.2.1 The Need of Trustworthy AI

In the last decade, AI has become omnipresent in humanity's daily lives. However, with all the benefits AI has brought forth, it also introduced potential risks of danger. Obermeyer and Mullainathan [8] showed in 2019 that the algorithm used at that time for important healthcare decisions for over 70 million people in the United States (US) is racially biased leading to black people being less likely to get enrolled in the care management program and receive complex and intensive future health care. Similarly, A. Brackey [9] has shown that the Correctional Offender Management Profiling for Alternative Sanctions (COMPAS) algorithm used in US court systems to predict if a defendant would become a recidivist is racially biased. COMPAS predicted twice as many false positives for recidivism for black offenders (45 %) than white offenders (23 %). But racial bias is not the only issue recent AI models have been exposed. Amazon has been using a hiring algorithm since 2014 to review job applicants' resumes and find the top applicants that fit the job profile. By 2015, the company noticed that the tool used for hiring was not gender-neutral in job roles like software engineers and architects and favored men over women. Investigating the root cause of this problem, engineers found out that the bias was introduced by the dataset used for training the algorithm. Since men are more prevalent in these job areas, the algorithm falsely learned that men are more suitable for these jobs than women [10].

Even if all the bias in the aforementioned AI systems has been mitigated, more examples of biased, harmful, or unsafe AI are found again and again. This shows the need of developing AI in a safe and trustworthy way to prevent unfairness, danger, and harm to humanity introduced by AI systems in the first place. This led to a lot of research conducted by researchers and governing bodies to propose definitions of trustworthy AI and guidelines to build trustworthy AI.

2.2.2 European Commission Ethical Guidelines for Trustworthy AI

The high-level expert group on artificial intelligence (AI-HLEG) [12], an independent expert group set up by the European Commission, presented a first draft of ethics guidelines for trustworthy AI in 2018, which were finalized and published in 2019. These ethics guidelines define three components for trustworthy AI, which should be met throughout the system's entire life cycle: "i) It should be lawful, complying with all applicable laws and regulations; ii) It should be ethical, ensuring adherence to ethical principles and values; and iii) It should be robust, both from a technical and social perspective" [12, p. 5].

Under these main principles, four ethical principles are defined: ”i) Respect for human autonomy, meaning that humans interacting with AI systems must be able to keep full and effective self-determination over themselves; ii) Prevention of harm, meaning that AI systems should neither cause nor exacerbate harm or otherwise adversely affect human beings; iii) Fairness, meaning that AI systems must ensure that individuals and groups are free from unfair bias and iv) Explicability, meaning that the processes, capabilities and the purpose of an AI system needs to be transparent” [12, p. 13]. In terms of the realization of trustworthy AI, the AI-HLEG declared the following seven requirements that must be met:

”**1. Human agency and oversight**, including fundamental rights, human agency and human oversight

2. Technical robustness and safety, including resilience to attack and security, fall back plan and general safety, accuracy, reliability and reproducibility

3. Privacy and data governance, including respect for privacy, quality and integrity of data, and access to data

4. Transparency, including traceability, explainability, and communication

5. Diversity, non-discrimination and fairness, including the avoidance of unfair bias, accessibility and universal design, and stakeholder participation

6. Societal and environmental well-being, including sustainability and environmental friendliness, social impact, society and democracy

7. Accountability, including auditability, minimisation and reporting of negative impact, trade-offs and redress” [12, p. 14].

The AI-HLEG further points out possible tensions between the principles and that methods of accountable deliberation to deal with such tensions should be established [12].

2.2.3 The Five Pillars of Trustworthy AI

Li et al. [24] and Liu et al. [25] have summarized the five main aspects of trustworthy AI: robustness, privacy, fairness, explainability, and accountability.

Robustness

An AI system is considered robust if it is able to deal with execution errors, erroneous inputs, or unseen data. The robustness is further categorized and summarized by Li et al. [24] into i) Data, ii) Algorithms, and iii) System robustness. Data robustness refers to the fact that the AI system must be trained considering the diverse distributions of data in different scenarios. The problem of distributional shifts became more prevalent with the widespread application of AI systems and the diverse environments in which the models are deployed [12]. A lack of data robustness may lead to a significant drop in the performance of the AI system. Algorithmic robustness refers to the fact that the algorithm is not vulnerable to adversarial and malicious attacks [26]. Decision-time attacks try to mislead the prediction of a given model by perturbing input samples in order to impersonate victims, poisoning attacks change the system's response to specific patterns by injecting carefully designed samples and model stealing attacks try to steal knowledge about the model, just to name a few. System-level robustness refers to the fact that an AI system should not allow illegal inputs. For example, presentation attacks may fool biometric systems by faking input photos or masks.

Another aspect of the robustness of an AI system that is defined by Li et al. [24] is the generalization. It refers to the capability to make accurate predictions on unseen data that share the same distribution as the training data [27]. Having a generalizable AI system is important for its trustworthiness since it is unreasonable to train the model on exhaustive amounts of data. Assuming a ransomware detection AI system that was trained and evaluated on ten different ransomware samples and sold to customers with the promise that the model has an accuracy of 97 % in detecting ransomware. The knowledge that the model generalizes to other ransomware samples is crucial, otherwise in practice many attacks will possibly not be detected and the trust in the AI system will diminish.

Privacy

Data that is used to train ML models hold a lot of sensitive data like names, gender, fingerprints, health records, and corporate information [18]. Rocher et al. [28] have shown that 25 demographic attributes were sufficient to identify 99% of the entities unique. Thus, AI models must protect against unauthorized use of data that can directly or indirectly identify a person or an organization and allow it to make inferences on the entity's private information.

Fairness

As introduced, AI systems are used in many fields such as financial risk assessment, hiring, and identification and already take decisions that impact human lives [29]. Systematic unfairness in such systems might have negative social ramifications and might put certain groups at a disadvantage. It is therefore essential for trustworthy AI to be fair and not biased in any direction. Bias in AI systems may have different causes, from data bias to model bias and procedural bias, and often manifests itself in the form of unfair treatment of different groups of people based on their protected information (e.g. race, ethnicity, religion, sexual orientation, gender, etc.) [30, 31]. Li et al. [24] categorized fairness of trustworthy AI into distribute fairness, meaning fairness of the outcome, procedural fairness, meaning fairness of the process, and statistical fairness, meaning the aggregated behavior of an AI system and individual fairness. Three different types of fairness have been identified by Caton et al. [32] at a group level: i) independence, which requires the system outcome to be statistically independent of sensitive variables such as gender or race. ii) separation requires that the independence principle holds, but is conditioned on the underlying ground truth. iii) Sufficiency requires that the true outcome and the sensitive variable are independent but similarly consider the ground truth.

Explainability

The extensive use of AI systems and their impact on humans create a demand among users for the right to know the intention, business model, and technological mechanism of AI products [12]. The understanding of how an AI model makes its decisions is summarized by Li. et al. [24] in the explainability pillar of trustworthy AI. Despite explaining the decision process in ML models has been an active topic in research [33, 34] the definition remains still an open question. Currently, studies divide the explainability of AI systems into two levels:

- Model explainability by design: A model is considered explainable if it is explorable by mathematics. Further, the model's complexity, in terms of the number of parameters is influential [14]. ML models that are recognized as explainable compromise linear regression, logistic regression, decision trees such as random forest, decision rules, and Bayesian and k-nearest neighbors (KNN) models [35].

- Post-hoc model explainability: Complex models such as DNNs have been shown to achieve better performance in industrial AI systems. However, relevant approaches still cannot fully explain these complex models. That is why post-hoc methods to explain models are an important aspect. These methods include analyzing the model's input, intermediate result, and output. Also, using explainable ML models, i.e. explainers, such as linear models are used to approximate the decision surface of complex models.

Accountability

AI systems need to follow regulations and law requirements in order to be trustworthy. This principle is summarized by Li et al. [24] in the accountability pillar of trustworthy AI. AI stakeholders must justify and document the architecture and implementation of their AI models and make this information accessible. This also implies that an AI system must be reviewed, assessed, and audited through the entire lifecycle [36].

Chapter 3

Related Work

This Chapter provides a literature survey of related work of trustworthy FL and its pillars in Section 3.1. It then continues with existing trustworthy FL taxonomies in Section 3.2 and tools to evaluate the trustworthiness of FL systems in Section 3.3 including comparing the survey to the existing solutions, consequently defining limitations. In Section 3.4 related works to estimate carbon emissions of AI/FL models are explored and Section 3.5 concludes this Chapter by providing findings from the related work.

3.1 Trustworthy FL - A Survey

The five pillars of the trustworthiness of AI systems, introduced in the Background Chapter, do also apply to FL. However, with FL being a distributed system including multiple clients, differences and additional requirements apply. This section provides a survey of related work to trustworthy FL.

3.1.1 Privacy

FL has been designed with the goal to collaborate on training a model and benefit without having to share information about the own private data. It is clear, that preserving the privacy of client's data is an important factor for an FL system to be trustworthy. Even though FL provides privacy to a certain degree by design, since the client's data is never shared but only the model parameters, a series of works have shown that attacks can still compromise data privacy [37, 38, 39]. Privacy attacks on FL come from either a dishonest server, a malicious client trying to infer (other) clients' private data, or an adversary in the middle that eavesdropped on the communication. Lyu et al. [40] provide an extensive survey on privacy attacks in FL. In membership inference attacks (MIA), attackers infer whether a specific given data exists in a client's private dataset [41, 37, 39]. In the training data/label inference attack (TIA) the attacker is able to reconstruct privately owned data samples through deep leakage from gradient (DLC)[38] and improved deep leakage from gradient (iDLC) [42] methods. Class representative attacks (CRA) aim to infer representative samples of a specific class using GANs [43, 44], whereas in property inference attacks (PIA) attackers aim to infer dataset properties, which may or may not be sensitive properties for example gender or age [37].

Similarly to the different privacy attacks that have developed over time, also privacy-preserving mechanisms have been explored. Lyu et al. [40] surveyed existing privacy-preserving methodologies which include i) Homomorphic encryption (HE), ii) Secure multiparty computation (SMC) and iii) Differential privacy (DP).

HE allows a robust solution by allowing arithmetic operations to be directly performed on ciphertexts and different forms of HE have been proposed [45, 46, 47] each having their advantages and disadvantages. In HE-based FL aggregation, local models are encrypted and aggregation is performed over ciphertexts, thus removing the need for a trusted server. However, HE introduces an overhead of commonly 15x increase in both computation and communication [48] making it unpractical for most real-world applications. However, Jin et al. [48] recently proposed FedML-HE, an efficient HE-based FL System that used optimization techniques and reduces the overhead introduced by HE significantly.

SMC enables different participants to perform a collaborative computation on their inputs without showing the inputs to other participants. Different SMC-based schemes have been proposed for privacy-preserving learning [40].

DP offers computationally effective privacy protection through perturbing the data but comes at the cost of accuracy. The different DP schemes include centralized differential privacy (CDP), where the trusted aggregator adds noise. In CDP, fewer perturbations are added and thus, the impact on the accuracy of the model is smaller but it is susceptible to attacks from the server, as well as attacks from adversaries coming from the outside [49]. In local differential privacy (LCP), clients perturb their data before sharing it, which offers more privacy but comes at cost of accuracy. Distributed differential privacy (DDP) closes the gap between LDP and CDP while ensuring the privacy of each individual by combining with cryptographic protocols [40]. Al-Ars and Enthoven [50] also surveyed privacy attacks and defenses in FL. Additionally to the HE, SMC, and DP they also included robust aggregation, gradient subset, gradient compression, and dropout as privacy-preserving technologies. Out of those, however, only dropout, a method generally used to prevent overfitting, can offer protection against some of the common attacks as can be seen in Table 3.1. For more detailed information on the attacks, defenses, and their functionality, please refer to [50, 40].

3.1.2 Robustness

The robustness of AI systems refers to the system being technically robust to ensure that they can not be maliciously used or bring harm to humans.

		Attacks			
		DLG/iDLG	MIA	mGAN-AI	GAN
Defenses	DP	+	+	*	+
	SMC	~	~	+	-
	HE	+	+	+	-
	Gradient Subset	~	+	~	~
	Gradient Compression	-	~	~	~
	Dropout	-	+	~	+
	Robust aggregation	-	~	~	-

Table 3.1: Privacy Attacks vs. Defense Mechanism in FL from [50].

Adversarial Robustness

Research has shown, that FL systems are vulnerable to both data poisoning [51, 52] and model poisoning attacks [53, 54]. Both together are summarized under the notion of poisoning attacks and they attempt to modify the behavior of the target model in some undesirable way. In data poisoning attacks, one or multiple clients modify their data such that the gradient arising from it and that is shared with the server is untrue and influences the global model in a wrong way. On the contrary, in model poisoning attacks the data is not touched, but the local model parameters are changed directly. It has been shown [53, 55, 54] that data poisoning attacks are generally less effective in FL due to the heterogeneous architectures. Despite this, they are often used because they are easy to implement. Further, poisoning attacks can be classified into i) untargeted poisoning attacks and ii) targeted poisoning attacks depending on the attacker's objective. In untargeted poisoning attacks, the goal is to randomly compromise the predictions of the target model. For example in Byzantine attacks, malicious gradients are sent to the server to cause failure of the global model [49]. In targeted attacks, the goal is that the global models output the target label specified by the adversary for a particular sample [56]. For example, in an FL setting that learns a model to distinguish between spam and non-spam emails, an adversary can implement a label-flipping attack, where it falsely labels all the data samples describing emails from a known spammer as not spam. The local model in turn then learns internally, that these emails are not spam and propagates this learning into the global model in the aggregation phase of FL [40].

Similarly to various poisoning attacks that have evolved over time, different defense strategies have been implemented. Defenses against targeted attacks include detection algorithms and erasing strategies. Defense against untargeted attacks mostly works by clustering or computing distance metrics like Euclidean distance, coordinate-wise median, and geometric median to separate benign from malicious clients. Given the functionality of these defense mechanisms, they only work if the majority or more of the clients are benign. Lyu et al. [40] surveyed and described state-of-the-art poisoning attacks and defenses in machine learning. Please refer to their paper for more detailed information on the functionality of poisoning attacks and defenses.

Algorithmic Robustness

For an FL system to be robust, its trained model needs to predict reliably and with high confidence, such that users can trust its output. The performance of a model is measured by its accuracy in predicting the test set. In FL, measuring the accuracy of the global model on the test set of the server side gives a good indication of the performance of the ML model overall and its convergence. However, in FL clients often have different local data distributions and the global model might not be able to patterns of each of them, to generalize well [57]. Therefore, it is essential to also look at the performance of the global model on the client's test data and its distribution [58]. Non-IID training data and poor generalization on the client side is a known problem in FL and many mitigation strategies, summarized under the notion of personalized FL, have evolved over time. There are two main personalized FL strategies defined by literature i) global model personalization and ii) learning personalized models. In global model personalization, the single trained global model is personalized for each client by additional training on each local dataset [59, 60]. In the learning personalized models strategy, instead of training a global model, many individual personalized FL models are trained. Architecture-based approaches aim to provide personalized model architectures for each client, such that the model architecture is suitable to capture the client's data, and similarity-based approaches aim to group similar clients and train models per client group [58]. In each personalization method, many sub-approaches and methodologies exist. For more details on all FL personalization methods, their advantages, and disadvantages, please refer to the comprehensive summary by Tan et al. [58].

Client Reliability

Since in FL, the global model is the result of aggregation of many client models, the client's reliability plays a huge role in the robustness of the FL system. Different works defined metrics and methodologies to measure clients' reputation and they are based on the quality of client's data [61], their contribution quality to the global model [62], their participation rate in the FL process and their computation resources [63] or a combination of the aforementioned [64]. Measuring client reputations is also used to detect malicious clients or free-riders [65]. However, measuring client's data quality and reputation is a critical aspect in FL, since evaluating those leaks of private information about the clients by definition, is not a preferred situation in FL. Therefore, there is a clear trade-off between the client's privacy and measuring the client's reliability to ensure a robust FL system. If the robustness or privacy of clients is more important for an FL system, depends on the context and the use case. Kang et al. [64] implemented a reputation-aware and reliable FL system that claims to be privacy-preserving by introducing a trusted third party on a blockchain that computes and holds the reputation score of the clients. Other than the client's reputation and data quality, also the dropout rate is considered crucial for FL systems. The more clients drop out, the less reliable the system is. Client dropout may have different reasons, under them connectivity issues or clients leaving the federation.

3.1.3 Fairness

Since FL models are trained over distributed devices with possibly different data quality and quantity and resources, their contribution to the final FL model may vary. In traditional FL approaches, each client receives the same global FL model independent of their contributions. This, however, may rise unfairness and may also hinder clients with high data quality and quantity to join FL federations. To address this issue, lots of research has been invested in fairness-aware federated learning (FAFL). Shi et al. [66] provides an extensive summary of introduced FAFL methods and fairness issues in FL and a FAFL taxonomy. They identified three key stages where unfairness may arise in FL systems.

Firstly, unfairness can occur in the stage of client selection. Most methods for client selection focus on the server's interest in increasing the convergence speed [67, 68, 69, 70] or enhancing the models' performance [71, 72]. Applying this client selection strategy can result in excluding clients with weaker capabilities from the FL process. This aspect is generally referred to as client selection fairness. Unfair client selection may also introduce bias in the FL model [73], for example by underrepresenting data coming from clients living in urban areas with weaker connections or by excluding devices with weaker capability.

Secondly, unfairness may arise during the optimization of the FL mode. Performing global optimization by aggregating clients' model updates may lead to the global model not being able to capture the diversity of the entire distributed training set, leading to prediction errors of the global model on client's datasets [74, 75]. This aspect is called performance fairness.

Thirdly, unfairness may occur in the incentive distribution stage. In traditional FL settings, each client is rewarded with the same global model neglecting their contribution. This might not be recognized as fair by clients who contribute more to the final FL model performance and may hinder them to participate in the federation, which in turn harms the model. This is commonly referred to as the free-rider issue [76]. To mitigate this fairness issue, FL systems need to reward clients based on their contribution to compensate for their expenses which is generally referred to as contribution fairness.

Further, same as in traditional AI systems, FL systems are considered fair if the resulting model is free from bias and does not discriminate against any group of individuals. Bias mitigation is more difficult in FL than in traditional centralized ML since depending on the privacy constraints of the FL system, the client data as well as the protected attributes are not known. Even worse, bias in FL is exacerbated since each party will introduce its own bias to the global model. Further, data heterogeneity is very common in FL systems and favors the construction of biased models [59]. Abay et al. [77] proposed local reweighting as a bias mitigation technique in the pre-processing stage of FL. It works by attaching weights to samples in the training dataset from each client locally, and thus does not come with privacy loss and is applicable regardless of the chosen ML model. However, it may come at a performance cost. They further propose an in-processing bias mitigation method called prejudice remover for FL, which works by adding a fairness-aware regularizer to the loss function. A disadvantage of the prejudice remover for FL is that it is tied to the logistic loss function and it requires clients to share their sensitive attributes.

3.1.4 Accountability

Accountability is important for the development of trustworthy and ethical AI systems. It means being responsible for the actions taken during the development and deployment of the FL system [12]. Further accountability means adhering to laws that are defined. To achieve accountability in FL, it is necessary to document the entire process such that questions can be answered. The documentation should include information about the participants, the data used, and the model's configuration, project specifications, and performance.

IBM Research has developed an accountable FL fact sheet [78] template that provides comprehensive documentation of the FL process, ensuring transparency and building trust in the system.

Besides documentation, monitoring is also a crucial aspect of accountability. Stakeholders must ensure that the AI system is developed and deployed according to the intended processes [12]. Auditing is commonly used to verify the accountability of AI systems. It involves evaluating the system's performance, verifying the documentation, and ensuring that the system adheres to ethical standards and the law.

3.1.5 Explainability

Similar to traditional AI systems explainability, explainability in FL refers to the ability to understand and interpret the decision-making process of the machine learning models trained using the FL approach. However, understanding how to model arrives at its predictions can be more challenging in FL, since the model is trained on data coming from different clients [79].

One aspect of explainability is interpretability, which refers to explainability from the model design. Interpretability is influenced by the models' transparency and the model size [25]. ML/DL models have been categorized by Arrieta et al. [35] by their interpretability level. A model is considered transparent if its behavior can be fully explained by domain experts and mathematics. In summary, algorithmic transparent models are Linear/Logistic Regression, Decision Trees, K-Nearest Neighbors (KNN), Rule-Based Learners, General Additive Models, and Bayesian Models. The non-transparent models include tree ensembles, multi-layer neural networks (MNN), convolutional neural networks (CNN), and recurrent neural networks (RNN) [35]. The definition of model size differs for different models. For instance, it can be the depth of a decision tree, the number of parameters for neural networks, the number of decision rules, and so on [80]. Explaining the relationship between the input and the output of the model becomes harder, the bigger the model size is [35].

3.1.6 Federation

The federation pillar of trustworthy FL has first been introduced by Sánchez et al. [14] and a prior master thesis [79] from the Communication Systems Group at the University of Zurich. Unlike a traditional ML model, an FL model is an entirely distributed system that comes with architectural design challenges. The federation pillar focuses on the interactions of different components of the FL system. It can be hard to interrelate the learning process amongst thousands of devices and still ensure model integrity and security [79]. Lo et al. [81] presented a collection of design patterns for the life cycle of an FL system.

The first important aspect is client management, FL systems need to attract clients to participate in the federation. Then, metadata needs to be collected about the clients such that the server can broadcast messages and start the training process [79]. Managing and verifying client information is typically done through a client registry. In CFL, the client registry is located on the server side and it collects information from clients such as device ID, connection uptime, computation power, and storage capacity. A client registry enables the FL system to keep track of clients and their status and is thus an important part of an FL system. During training, the server needs to select participants in each training round. This can be done at random or through a client selector [79]. Selecting clients at random may not fully exploit the local updates from heterogeneous clients, resulting in lower model accuracy, slower convergence rate, degraded fairness and more [82]. The client selector may choose clients based on different metrics and different client selectors have been proposed through time, each having their own advantages and disadvantages.

Since in FL, a global model and numerous local models are trained through many rounds, model co-versioning is an important strategy to ensure traceability and fallback in case something goes wrong or if an attack happens that degrades the model. A co-versioning system might be integrated on the server side that stores global model versions [79]. However, needless to say, storing all these models comes at high storage cost [14].

3.2 Existing Trustworthy FL Taxonomy

Table 3.2 summarizes the existing trustworthy FL taxonomies and their coverage of trustworthy FL pillars and of the seven requirements defined by the AI-HLEG. The taxonomy from Shi et al. [66] covers the pillar of fairness and partially the federation pillar since it discusses fair client selection as discussed in Subsection 3.1.3. The taxonomy from Tariq et al. covers the pillar of privacy, fairness, and robustness and includes requirements number two, three, and five defined by the AI-HLEG. Liu et al. provided a taxonomy covering the pillar of privacy, robustness, and partially the pillar federation as shown in Subsections 3.1.2 and 3.1.1. The taxonomy that covers the most pillars and requirements defined by the AI-HLEG is the trustworthy FL taxonomy from Sánchez et al. [14]. Since this is the most advanced taxonomy, that covers six of the seven requirements defined by the AI-HLEG, it is discussed in more detail in the following.

Table 3.2: Existing Trustworthy FL Taxonomies and Their Coverage of Pillars and AI-HLEG Requirements

Authors	Paper	Pillars / AI-HLEG Requirements						
		Privacy	Fairness	Robustness	Accountability	Explainability	Federation	Sustainability
		3. Privacy and data governance	5. Diversity, non-discrimination, and fairness	2. Technical robustness and safety	7. Accountability and auditability / 1. Human agency and oversight	4. Transparency including explainability	2. Technical robustness and safety / 5. Diversity, non-discrimination and fairness	6. Environmental well-being
Shi et al. [66]	A survey of fairness-aware federated learning	no	yes	no	no	no	partially	no
Tariq et al. [83]	Trustworthy Federated Learning: A Survey	yes	yes	yes	no	no	no	no
Liu et al. [25]	Threats, attacks and defenses to federated learning: issues, taxonomy and perspectives	yes	no	yes	no	no	partially	no
Sanchez et al.[14]	Learning: issues, taxonomy and perspectives	yes	yes	yes	yes	yes	yes	no

A previous master thesis [79] from the Communication Systems Group of the University of Zurich and the consequent paper published by Sánchez et al. [14] surveyed state-of-the-art literature to trustworthy FL and created a taxonomy describing pillars, notions, and metrics to define trustworthy FL requirements. The taxonomy is depicted in Figure 3.1 and contains the pillars i) privacy, ii) robustness, iii) fairness, iv) explainability, v) accountability, and vi) federation. For each pillar, notions and according metrics are defined. In total, 36 metrics are defined that can be used to evaluate the trustworthiness score of a given FL system. For more detailed information, please refer to [14].

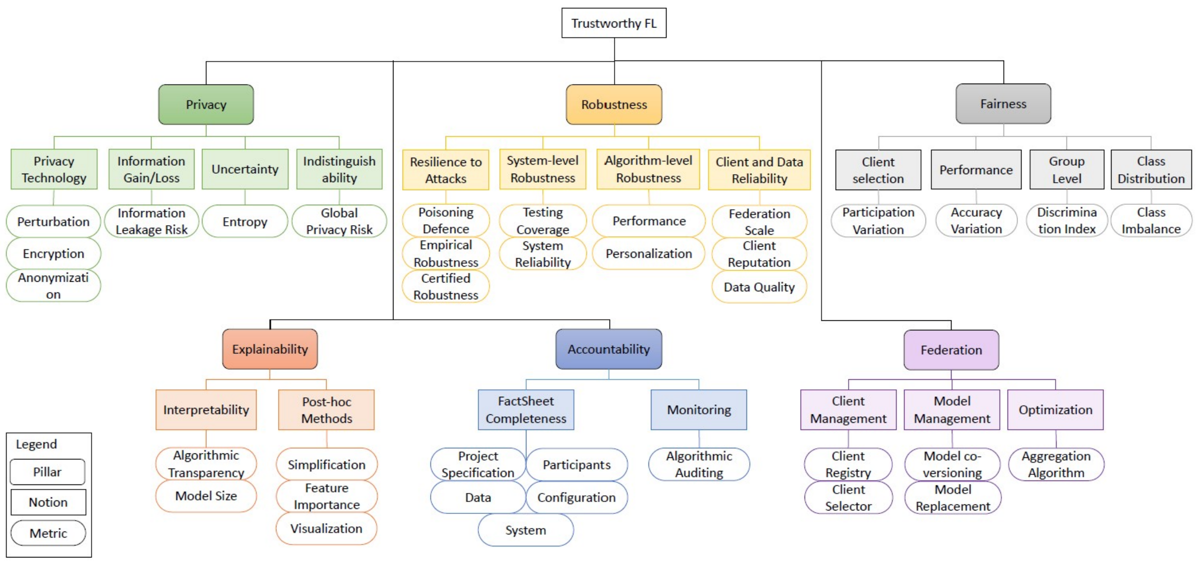


Figure 3.1: Trustworthy FL Taxonomy from [14].

3.2.1 Limitations

Comparing the existing trustworthy FL taxonomy and the extensive literature survey, despite the considerable work and depth of the taxonomy, limitations become present.

The most important limitation becomes present when comparing the taxonomy to the requirements defined by the AI-HLEG and the existing taxonomy. The environmental impact of an FL system is not considered in the taxonomy, but environmental well-being has clearly been defined as one of the seven requirements for trustworthy AI by governing bodies [12] as discussed in Subsection 2.2.2. Emissions produced by AI gain more and more attention from research since the ever-growing AI models created an ever-increasing carbon footprint that can not be neglected in a time where global warming is the key challenge of humanity.

In the robustness pillar client dropouts should be considered as well since high dropout rates threaten the robustness of a system. Further, only the accuracy of the global model on the global test set is considered as the performance metric of the FL system, but to measure the performance trustfully the performance on the test set of clients (local) should be considered as well to ensure the collaboratively trained model also performs robust on clients datasets.

In the fairness pillar, the collaboration fairness aspect, describing that rewards should be aligned with contribution, is missing completely but is important to attract high-quality clients, and rewards based on contribution have been shown to contribute to the robustness of FL systems as well.

3.3 Evaluation Tools of Trustworthy FAI

In terms of tools evaluating the trustworthiness of a federated learning system, Sánchez et al. [14] implemented FederatedTrust. FederatedTrust is a lightweight algorithmic prototype, that evaluates the trustworthiness score of a FL system by scoring it according to measurements in the six defined pillars i) privacy, ii) robustness, ii) fairness, iv) explainability, v) accountability, and vi) federation. The 20 metrics implemented by the prototype are depicted in Table 3.3. The prototype is programmed as a Python package called FederatedTrust that is supposed to be integrated into the FederatedScope environment. FederatedScope is a Python environment to develop and test FL systems. A more detailed view of FederatedTrust v.0.1.0 is provided in Subsection 5.1.3.

Table 3.3: Metrics Implemented by the FederatedTrust Algorithm Prototype from [14].

Metric	**Description**	**Input**	**Output**	**When**	**Who**
Privacy					
Differential Privacy	Use of global or local differential privacy as a privacy defense	FactSheet	0/1	Pre-training	Server
Entropy	Uncertainty in predicting the value of a random variable	FL Framework Conf	[0, 1]	Pre-training	Server
Global Privacy Risk	Maximum privacy risk with differential privacy based on e	Client Statistics	%	Pre-training	Server
Robustness					
Certified Robustness	Minimum perturbation required to change the neural network prediction	FL Model	Real	Post-training	Server
Performance	Test accuracy of the global model	Statistics, FL Model	%	During-training	Clients
Personalization	Use of personalized FL techniques	FactSheet	0/1	Pre-training	Server
Federation Scale	Number of clients representing the scale of the federation	FactSheet	Integer	Pre-training	Server
Fairness					
Participation Variation	Uniformity of distribution of participation rate among clients	FL Framework	[0, 1]	Post-training	Server
Accuracy Variation	Uniformity of distribution of performance among clients	Client Statistics, FL Model	[0, 1]	During-training	Clients
Class Imbalance	Average class imbalance estimation among clients	Client Statistics	[0, 1]	Pre-training	Clients
Explainability					
Algorithmic Transparency	Interpretability of the model by design	FL Model	[1, 5]	Pre-training	Server
Model Size	Model Features dimensionality, depth of decision tree, or number of parameters, number of features	FL Model	Integer	Post-training	Server
Feature Importance	Average variance of feature importance scores	FL Model	[0, 1]	Post-training	Server
Accountability					
Project Specification	Project details and purpose	FactSheet	0/1	Pre-training	Server
Participants	Participants number, identifiers, and their organizations	FL Framework Conf, FactSheet	0/1	Pre-training	Server
Data	Contains Data origin and data-preprocessing steps	FL Framework Conf, FactSheet	0/1	Pre-training	Server
Configuration	Information about the FL model	FL Framework Conf, FactSheet	0/1	Pre-training	Server
System	Contains training time, FL model size, and network performance	FL Framework Conf, Statistics	0/1	Post-training	Server
Federation					
Client Selector	Use of a client selector scheme rather than random selection	FactSheet	0/1	Pre-training	Server
Aggregation Algorithm	Selected aggregation function	FL Framework Conf	%	Pre-training	Server

3.3.1 Limitations

As already discussed in the limitations of the formal taxonomy, the prototype is also missing a pillar including notions and metrics representing the environmental impact of a FL system as part of its trustworthiness score, despite it being defined as a key requirement for trustworthy AI. Since it is a first, lightweight prototype, there are also other aspects not implemented.

The current privacy-preserving technologies considered in the prototype only evaluate the implementation of differential privacy. However, the parameters used for differential privacy are crucial to determine its usefulness but vary from system to system. Furthermore, other encryption-based privacy-preserving technologies like homomorphic encryption (HE) and secure multiparty computation (SMC) can offer privacy protection as well. Neglecting these technologies can lead to the misclassification of FL models in the FederatedTrust algorithm, such as models having HE implemented being incorrectly labeled as untrustworthy. Scoring privacy-preserving technologies with a binary 1 or 0 based on their implementation is too simplistic. Further, the information leakage risk is not considered in the prototype at all, but may combined with privacy-preserving technologies capture the pillar well. A more sophisticated approach would be to score different privacy-preserving technologies based on their effectiveness against common privacy attacks identified in the FL setting.

The robustness pillar does not include the countermeasures to attacks, called poisoning defense, and many of them have been proposed as discussed in the survey of the robustness pillar in Subsection 3.1.2. The system-level robustness notion could be implemented as well. For client and data reliability, client reputation and clients data quality metrics could be implemented. However, this would leak the private information of the clients which a trustworthy FL quantifier should not do.

In the fairness pillar, the notion of collaborative fairness is not considered. Further, group-level fairness is missing and could be represented by the existence of implemented bias mitigation strategies.

3.4 Estimating Emissions of AI/FL

As shown in Table 3.2, to the best knowledge, none of the existing trustworthy FL taxonomies has the environmental impact requirement of the AI-HLEG [12] integrated. This creates a gap between methodologies applied by researchers and requirements set by governing bodies. Thus, this section explores related work to estimate the environmental impact of AI/FL models. Despite measuring or estimating the influence AI has on the environment is relatively new, in recent years few works have tried to address this problem.

Most works focus on estimating the carbon emissions of specific models and Lucconi et al. [84] provided a survey on aspects that influence the CO_2 emissions of ML. Strubell et al. [85] estimated the financial and environmental costs of large natural language processing (NLP) models by analyzing the training and fine-tuning process. Luccioni et al. [86] estimated the carbon emissions of the large language model BLOOM having 176 billion parameters to be 50.5 tonnes of CO_2 equivalents. Patterson et al. [87] estimated the energy consumption and computed the carbon emissions of the language models T5, Meena, GShard, Switch Transformer, and GPT-3 and highlighted opportunities to improve energy efficiency and CO2 equivalent emissions such as sparsely activated DNNs and using energy grids with low carbon intensity.

While the mentioned works focus mainly on energy consumption, George et al. [88] point out that water consumption to cool large data- and server centers also contributes heavily to the environmental impact of AI models and estimated the water consumption needed to run Chat-GPT.

Despite most works focusing on centralized machine learning model training, Qui et al.[89] provided a first look into the carbon footprint of FL models by incorporating parameters that are special to FL and comparing the emissions produced by FL models vs. emissions produced by centralized ML models. They concluded, that FL models can emit up to two orders of magnitude of CO_2 equivalent emissions if the data is not identically distributed, which is often the case in FL.

Similarly to estimating the carbon emissions of AI/FL models, tools to track carbon emissions and apply standardized measurements for better comparison of model emissions have been developed [86]. Code Carbon [90] and the Experimental Emissions Tracker [91] can be used to track emissions during the training process, while the ML CO2 Calculator [92] can be used to calculate the emissions after training.

Despite all the works done in this research field, to the best knowledge, none of them have incorporated the emissions produced by FL models into trustworthy FL despite environmental well-being clearly being defined as one of the seven key requirements for trustworthy AI/FL by the AI-HLEG [12].

3.5 Findings from Related Work

FL has gained popularity in recent years and extensive research has been conducted related to the different pillars of trustworthy FL. While early works have provided first trustworthy FL taxonomies, they have limitations in terms of neglected aspects that need to be considered in trustworthy FL. The most considerable limitation is the missing aspect of environmental well-being in the formal taxonomies as well as in the algorithmic prototypes. While all other six key requirements defined by the AI-HLEG are present, the environmental well-being aspect is missing and has to the best knowledge not been defined or considered by any trustworthy FL taxonomy. This creates a gap between methodologies applied by research and requirements set by governing bodies. Thus, besides the extensive survey of the pillars and metrics provided in this Chapter, this work introduces the environmental impact aspect to trustworthy FL, defines a pillar, notions, and metrics for sustainability in trustworthy FL, and implements it in an algorithmic prototype, such that trustworthy FL research adheres to the regulations defined by the AI-HLEG and raises awareness of AI developers to the important aspect of emissions produced by the training and use of FL models.

Chapter 4

The Sustainability Pillar of Trustworthy FL

As defined by AI-HLEG, for AI systems and thus also for FL systems to be trustworthy, environmental well-being is one of the seven key requirements [12]. However, no work defined and included the environmental impact of the FL system as part of its trustworthiness, leaving a gap between trustworthy FL research and requirements defined by governing bodies. Thus, this work contributes to the research community by defining a trustworthiness pillar for FL systems called sustainability paying attention to the environmental impact produced by the FL system, resulting in the first trustworthy FL taxonomy including all seven requirements defined by the AI-HLEG.

This chapter first describes the motivation and background for sustainability in trustworthy FL in Section 4.1 and then continues with the notions and metrics that have been chosen to define the sustainability pillar of trustworthy FL in Section 4.2, including the carbon intensity of the energy source in Subsection 4.2.1, the efficiency of the underlying hardware in Subsection 4.2.2 and the complexity of the federation in Subsection 4.2.3. It continues by discussing the limitations of the approach in Section 4.3 and concludes the Chapter with the novel trustworthy FL taxonomy including the defined sustainability pillar in Section 4.4.

4.1 Motivation and Background

Currently, the focus of AI is set on achieving ever higher accuracy and thus developing ever higher-performing models. However, tremendous amounts of resources are used to train and run these AI systems, which comes with a cost to the environment [84]. Ever higher accuracy is mostly but not only achieved by increasing the parameters, which in turn also increases the energy consumed to train these models [87]. By only looking at the training of the models, as an example, the large language model GPT-2 has 1.5 billion parameters and consumed 28'000 kWh of energy to train. The successor GPT-3 has 175 billion parameters and consumed 248'000 kWh of energy to train, which is 10 times more energy than GPT-2 [86]. This example illustrates the trend to ever-growing machine learning models that have an ever higher resource consumption. However, comparing the ML model's resource consumption by the energy used is not optimal, since the emission in terms of CO_2 equivalents (CO_2eq) is produced by these amounts of energy, depending on the energy mix used to train these models. Therefore, CO_2eq emissions are a rational metric to compare and evaluate the environmental impact of different AI systems. Patterson et al. [93] estimated the CO_2eq emissions of GPT-3 to be 552.1 tons of CO_2eq. To put these emissions into perspective, an average Dane generates 11 tons of CO_2eq in a year, meaning that the carbon footprint of GPT-3 is roughly that of the annual carbon footprint of 50 Danes. This is a large carbon footprint, remembering that GPT-3 is just one of many AI systems. Therefore, tracking and estimating the carbon emissions for AI systems is a crucial step to take for developers and environmental well-being is a key requirement for trustworthy AI systems. No reports could be found that report the CO_2 equivalent emissions of FL systems in use. However, Qui et al. [89] tried to estimate the CO_2eq emissions of FL models and compared them to traditional centralized ML models. They concluded, that FL may emit up to two orders of magnitude more carbon than traditional centralized ML if the data is non-iid due to lower convergence speed and more training rounds. If the data is iid, then the carbon emissions of FL systems and traditional centralized ML are comparable.

Emissions produced by different ML and FL systems are commonly compared by the carbon footprint. The carbon footprint is calculated by Equation 4.1 where E is the number of electricity units consumed during the computation procedure quantified in kilowatt-hours (kWh) and C is the amount of CO_2eq emitted from producing one electricity unit. C is often quantified as kg of CO_2eq emitted per kWh of electricity and is sometimes referred to as the carbon intensity of the electricity [90].

$$Carbon footprint = E \times C \tag{4.1}$$

CO_2eq describes CO_2 equivalents and is a standardized measure describing how much warming a given amount of gas will have since other gases such as methane and nitrous oxide also have a warming effect and are emitted by producing electricity as well. As described, the carbon footprint is dependent on the electricity consumption and the carbon intensity of the grid used.

In FL, emissions may come from the training phase from the clients as well as from the aggregation at the server side. Further, emissions are produced by the communication costs coming from uplink communications (clients sharing the model with the server) and downlink costs (the server sharing the model with the clients). The amount of electricity that is used for training the FL model is dependent on many factors. First, the efficiency of the underlying hardware at the client's side and at the server side affects the consumed energy. Secondly, it depends on the federation's complexity and size, the bigger and the more complex the federation the more computational power is needed. The amount of carbon that is produced then additionally to the energy consumed depends on the carbon intensity of the energy grid that is used [87, 84]. Consequently, this thesis proposes two possible strategies for defining and evaluating the sustainability pillar of trustworthy FL:

1. Strategy one: Absolute measurement of emissions: directly measure/estimate and score the CO_2eq emissions produced for training, aggregation, and communication of an FL system.

2. Strategy two; Relative scoring of aspects that influence emissions: score the carbon intensity of the electricity grid used, the efficiency of the underlying hardware, and the complexity of the federation.

The pillars including notions of the two strategies are shown in Figure 4.1. The left side shows strategy one and the right side strategy two.

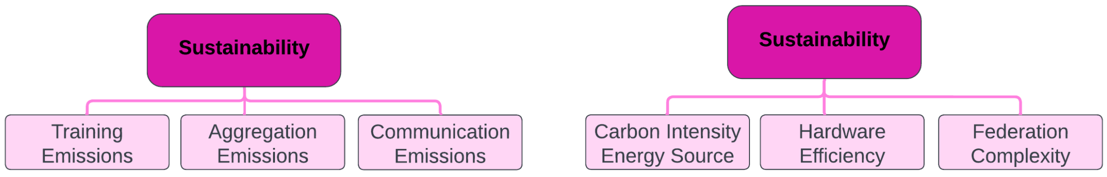


Figure 4.1: Two possible Strategies to define Sustainability Pillar Notions (left:absolute, right:relative).

Strategy one provides the advantage that absolute values of CO_2eq emissions are evaluated and scored. But it provides the disadvantage that it is unclear how to score the emissions produce answering the question of how much emissions are acceptable and how much are not. Further, it leads to the result that big and complex federations are always scored badly in the sustainability pillar and small and simple federations are scored well. Despite the fact that the complex federation may have taken the best possible decisions in terms of being as sustainable as possible, meaning using efficient hardware, and not carbon-intensive energy sources. Complex federations including big models and many clients are sometimes necessary to solve complex problems, and it is thus unfair to compare such federations to small federations that solve a much simpler task at hand. Similarly comparing the emissions from China to the emissions from Switzerland is not fair since much more people live in China. The second strategy provides the advantage that comparing completely different federations in a fair way is possible by also including the decisions that have been taken that influence sustainability, whilst still keeping the fact that big federations emit more emissions through including the federation complexity. Secondly, decisions that influence sustainability can be scored in a clear way since the ranges are given. Because of these advantages, the second strategy is followed in this thesis. The following section introduces the notions and metrics defined for the sustainability pillar of trustworthy FL.

4.2 Notions and Metrics of the Trustworthy FL Sustainability Pillar

This section discussed the notions and metrics that define the sustainability pillar of trustworthy FL.

4.2.1 The Carbon Intensity of the Energy Source

The carbon intensity of electricity is different in different parts of the world and depends on the energy mix that is used to produce electricity. The United Nations (UN) Intergovernmental Panel on Climate Change (IPCC) [94] has provided a median value of gCO_2e per kWh for different energy fuels as shown in Graph 4.2. Wind and nuclear emit the least CO_2eq with 12g and 11g of CO_2eq per kWh and coal the most with 820g of CO_2eq per kWh. This is a difference of factor 68. Thus, an FL system that has used 500 kWh of energy to be trained, would have emitted 5.5 kg of CO_2eq if it was trained on electricity produced by nuclear energy and 410 kg of CO_2eq if it was trained on electricity produced by coal only. This showcases, that the energy grid that is used to train a FL system plays a huge role in the carbon emissions produced. Similarly, the carbon intensity of the energy grid of countries varies by a huge factor.

British Petroleum (BP) has published in their annual review of the worlds energy statistics [95] that the least carbon-intensive energy grid is used by the African country Lesotho with 20g of CO_2eq per kWh and the most carbon-intensive energy grid is used by the South African country Botswana with 795 of CO_2eq per kWh in 2022. Switzerland is in place 12 of the countries with the lowest carbon-intensive energy grid with 32g of CO_2eq per kWh. The carbon intensity of the energy grid by countries is visualized in Figure 4.3.

Thus, the carbon intensity of the energy source notion of the Sustainability pillar deals with the carbon intensity of the energy grid used. As shown, it ranges from 20g of CO_2eq to 795 of CO_2eq by looking at the countries' energy grids. Theoretically, with the energy sources available today, the lowest possible energy grid would have 11g of CO_2eq per kWh only using wind energy and the highest possible 820g of CO_2eq only using coal energy. The energy grid used can be determined by the location of the workers (retrieved from the IP address). For the carbon intensity of the energy grid used by clients, the average of all the energy grids used by clients is computed. For the carbon intensity of the energy grid used by the server, the energy grid of the country the server operates in is taken.

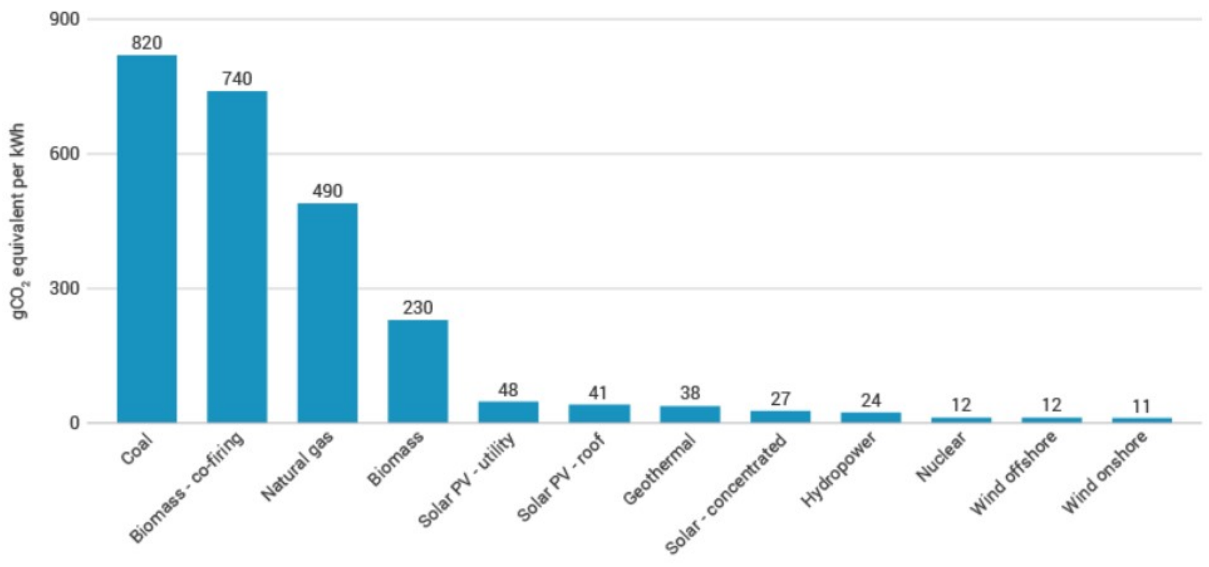


Figure 4.2: Average life-cycle CO_2e emissions per Fuel to Produce 1 kWh of Electricity from [94].

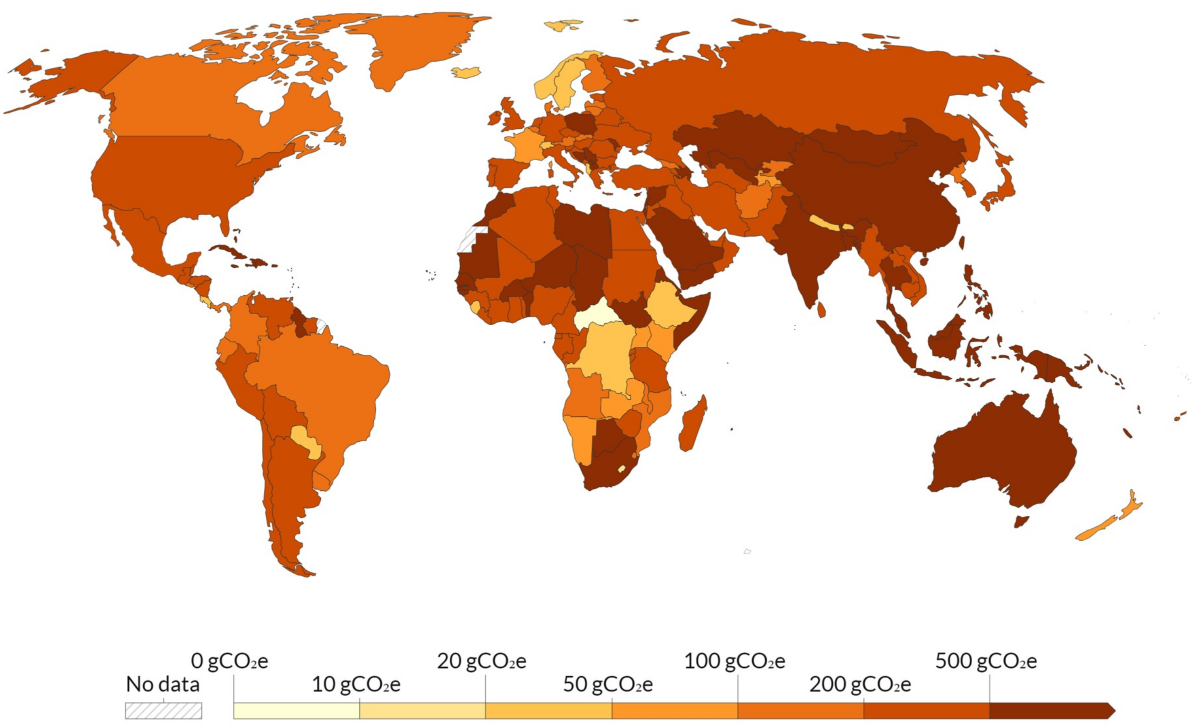


Figure 4.3: Carbon Intensity of Electricity in 2022 by Country from [96].

4.2.2 The Efficiency of the Underlying Hardware

The second aspect that significantly impacts the energy consumption and thus the emissions of an FL system is the efficiency of the underlying hardware. Efficient hardware consumes less power to perform computational tasks. Lower power consumption translates to reduced energy requirements, leading to lower CO_2eq emissions. In contrast, inefficient hardware that consumes more power will contribute to higher energy consumption and thus increased CO_2eq emissions. Additionally, inefficient hardware tends to generate more heat, necessitating additional cooling mechanisms such as air conditioning or fans. These cooling systems consume energy and contribute to CO_2eq emissions. By contrast, efficient hardware produces less heat and may require fewer cooling resources, leading to lower energy consumption and reduced emissions [92]. FL systems train ML models and are computationally heavy, thus the efficiency of the underlying hardware plays a significant role in the emissions that are produced by the FL system.

The performance of Central Processing Units (CPUs) and Graphics Processing Units (GPUs) can be described by different metrics, such as clock speed, Floating-Point Operations Per Second or Instructions Per Second (IPS) [97]. It is important to note that none of these metrics play a complete picture of the performance of the processing units, and different metrics are more applicable in certain use cases. Further, manufacturers of CPUs and GPUs often do not fully disclose the metrics of their products, which makes comparing them difficult. To solve this issue, lots of benchmarking software has evolved [98]. Such benchmarking tools evaluate the processor's performance across a range of tasks and provide a score that can be used for comparison. The most popular benchmarking software for processors is PassMark, it computes a performance score by running standardized tests that simulate real-world workloads, such as executing complex mathematical calculations. PassMark has collected the benchmarks of over a million computers and made them available to the public, allowing them to compare the performance of processors.

In terms of heat production of a processor, Thermal Design Power (TDP) is used as a specification in the industry. It indicates the maximum amount of heat a computer component, such as a CPU or GPU, is expected to generate under normal operating conditions. TDP is typically expressed in watts and represents the maximum power consumption and heat dissipation that can be expected under typical workloads. The smaller the number for TDP, the lower the power consumption of the processor [99].

To evaluate the efficiency of the underlying hardware in terms of computing power per unit of power consumed, it makes sense to divide the Mark through the TDP, defining the power performance of the processor. A processor with a high power performance score is able to do a lot of computation with low energy consumption and it is thus the most efficient in terms of resource consumption [99]. PassMark has provided a database with Power Performance measurement for over 3000 CPUs published on Kaggle [100] and for over 2000 GPUs [101], which will be used to evaluate the efficiency of the processors used by the server and the average efficiency used by the clients in this thesis's algorithmic prototype.

4.2.3 The Complexity of the Federation

The complexity and size of the federation impact the consumed energy thus the emissions produced. Generally, the more complex, the bigger the model, the higher the number of participants, and so on, the higher also the energy consumption [89].

Number of Training Rounds

Each training round in an FL system produces CO_2eq emissions by the energy that is used [89]. One training round consumed energy for i) training of the model on the client's side, ii) aggregating the model parameters on the server side, and iii) communication of model parameters up (from clients to server) and down (from server to clients). All of these consume energy and thus emit CO_2eq. More training rounds emit higher amounts of CO_2eq than fewer training rounds.

Dataset Size

The size of the dataset that is used by clients to train the FL models influences the CO_2eq emissions. Larger datasets need more computational resources in terms of power and memory and time to fit the model. Thus, larger datasets need more energy than smaller datasets and also produce more CO_2eq [92].

Model Size

The size of the model that is trained in the FL system influences the CO_2eq emissions produced as well. Large models typically require more computational resources and time to process each iteration, which results in higher energy consumption [92] at the client's side. Also, aggregating large models on the server side typically uses more energy than aggregating small models. Further, in FL, model parameters are shared between server and clients, large models thus also introduce a communication overhead, again leading to more energy consumption and more CO_2eq emissions.

Number of Clients

The number of clients in a federation, also called the federation size, influences the emissions produced by this system. The more clients participate in the federated, the more energy is used [89] for i) training, ii) aggregation, and iii) communication, and thus, the more CO_2eq are emitted.

Client Selection Rate

The client selection rate in an FL system refers to the number of clients that are selected to share their model parameters in each round with the server. Often, only a percentage of clients is selected per round [89]. The larger this percentage, the larger the communication overhead from the uplink communication. Needless to say, the larger the CO_2eq emissions.

Number of Local Training Rounds

In FL, clients train their local models and at some point in time the model parameters are shared with the server. However, clients may perform a different number of local training rounds within one global training round. The higher the number of local training rounds, the higher the computational overhead on the client's side and the higher the energy consumption [92, 89].

4.3 Limitations

For the carbon intensity of the energy source notion, a limitation is that taking the average carbon intensity of the energy grid of the country is just an approximation since the carbon intensity of the electricity grid fluctuates within a country and also within a day or within seasons. However, for the purpose used it is a fairly good approximation.

For the hardware efficiency notion, only the efficiency of CPUs and GPUs is considered, but of course, to be more accurate also the efficiency of other parts such as RAM could be integrated. The power performance metric is dependent on PassMark benchmarking scores and is not comparable to other benchmarking software scores. Further, if the true emissions want to be captured, the emitted CO_2eq for producing the hardware should be included. This, however, is fairly difficult to do.

Aspects that may be relevant for emissions produced but are not paid attention to in this first proof of concept sustainability pillar could be privacy-preserving technologies used in the federation. For example, if a federation uses HE as privacy protection, due to its computational complexity may increase the energy consumption needed and thus also the emissions produced. Further, as introduced in Subsection 3.1.2 FL systems often use methods to detect malicious clients or free-riders such as clustering or the H-MINE algorithm. Such methodologies are computationally heavy and may increase the computational costs, thus the energy consumption and CO_2eq emissions.

4.4 Trustworthy FL Taxonomy

The six pillars of trustworthiness defined by Sánchez et al. [14] together with the newly introduced pillar sustainability constitute a comprehensive taxonomy describing the requirements for a trustworthy FL model. Thanks to the newly introduced pillar sustainability, the taxonomy including seven pillars aligns with the seven requirements for trustworthy AI defined by the AI-HLEG [12]. A visual representation of the final taxonomy including the seven pillars of robustness, privacy, fairness, explainability, accountability, federation, and sustainability is presented in Figure 4.4. In each pillar, the most important aspects representing the pillar are summarized in notions [79], as discussed the sustainability pillar has three notions - Carbon intensity of the energy source, hardware efficiency, and federation complexity. Within each notion, specific metrics that can be used to calculate the trustworthiness level are defined. This taxonomy forms the builds the basis for evaluating and assessing the level of trustworthiness in an FL system. The taxonomy can be customized by adding or removing metrics, depending on the context of a system.

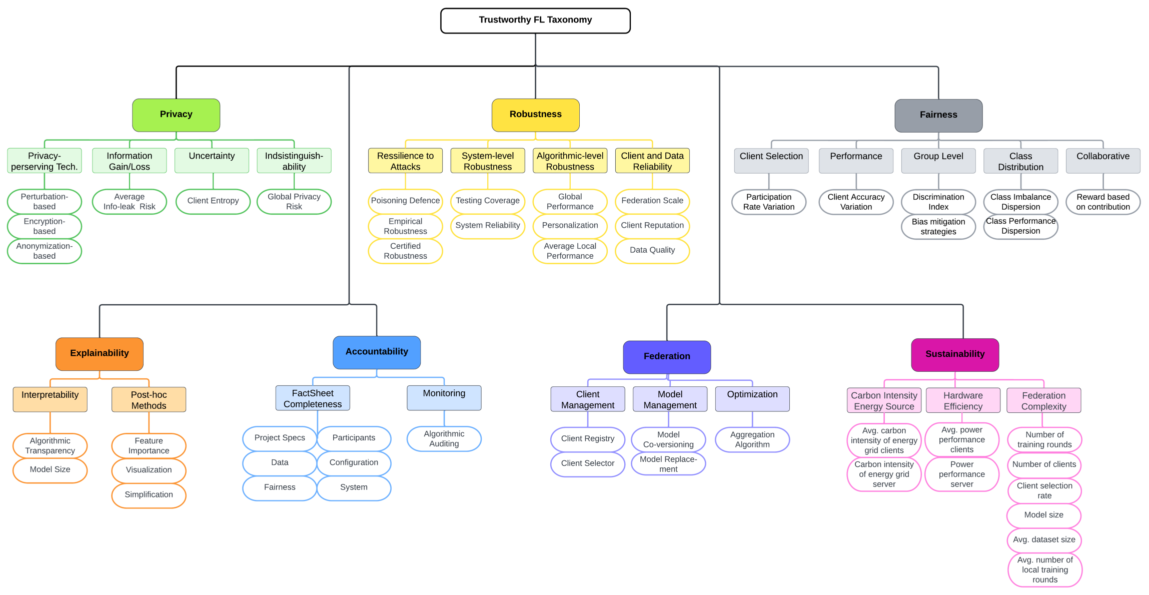


Figure 4.4: Trustworthy FL Taxonomy

Chapter 5

Design and Implementation

The second contribution of this work is the design and implementation of the sustainability pillar defined in Chapter 4 into an algorithmic prototype to evaluate the trustworthiness score of an FL system.

Even though this work has found the limitation of the missing aspects of paying attention to the emissions produced by an FL system as part of its trustworthiness, the goal of the algorithmic prototype still is to evaluate the overall trustworthiness of a FL system including all seven pillars. So far, no work has presented an evaluation algorithm for trustworthy FL that includes all the seven requirements defined by the AI-HLEG. Thus, instead of only implementing a prototype for the sustainability pillar, this work extends the already existing prototype FederatedTrust developed by Sánchez et al. [14] with the sustainability pillar and consequently presents the first algorithmic prototype to evaluate the trustworthiness score of a given FL system including all the seven pillars representing the seven requirements defined by the AI-HLEG [12].

Section 5.1 introduces tools that are used in the prototype. As FederatedTrust is implemented to be integrated into FederatedScope, this Chapter first gives an overview of FederatedScope and its functionalities in Subsection 5.1.1. It continues with an overview of FederatedTrust v.0.1.0, the existing version implemented and designed by [14] and a description of the python package CodeCarbon, which is used in the implementation to estimate hardware efficiency and the carbon intensity of the energy grid used in Subsection 5.1.2.

After introducing the tools used, the design and implementation of FederatedTrust v.0.2.0 are handled in Section 5.2.

5.1 Used Tools

This Section gives insights into the design, architecture, and important functionalities of tools that are used in FederatedTrust v.0.2.0. It starts with the Framework FederatedScope in Subsection 5.1.1, continues with the package CodeCarbon in Subsection 5.1.2 and ends with details to FederatedTrust v.0.1.0 in Subsection 5.1.3.

5.1.1 FederatedScope

FederatedScope [102] is an FL platform providing comprehensive functionalities and flexible customization for various FL tasks. It is based on an event-driven architecture, allowing users to extend and customize the FL system. It is written in Python and can be installed in a Python environment via the following command:

```
git clone https://github.com/alibaba/FederatedScope.git
```

FederatedScope has a modular construction and provides different interfaces where users can add new building blocks as shown in Figure 5.1. New workers (clients and server), new algorithms for personalization and/or aggregation and plugins for attack simulations, privacy protection strategies, and much more can be programmed and easily integrated into the framework [102]. This makes the framework extensible and adaptable to different use cases and an interesting and suitable framework for research in FL. In this work, version 0.2.0 of FederatedScope is used.

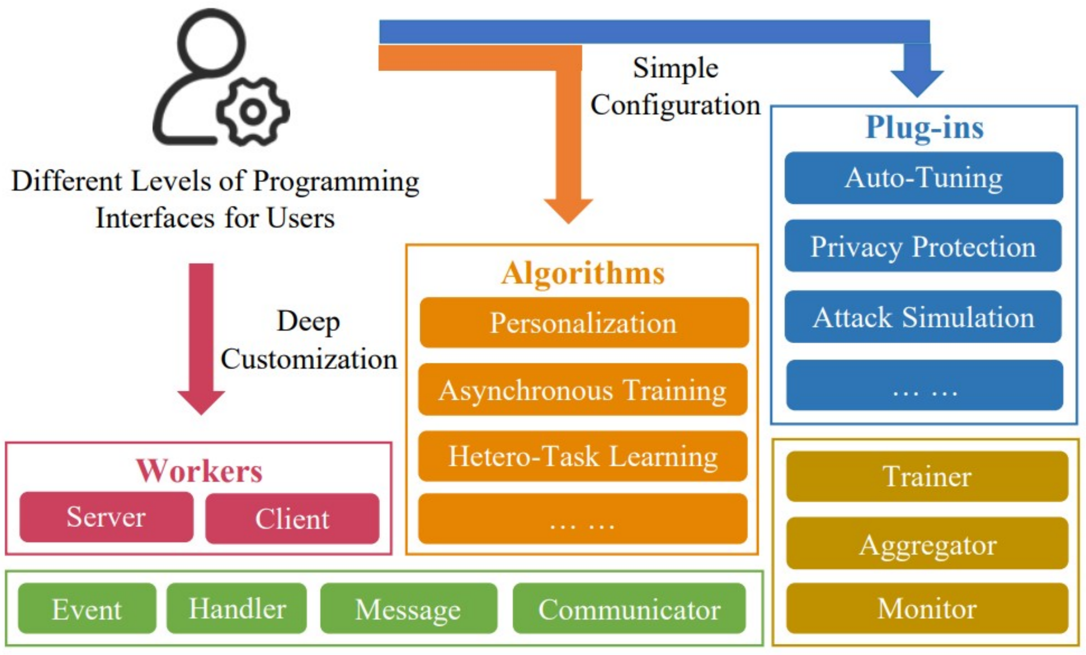


Figure 5.1: Programming Interfaces provided by FederatedScope from [102].

Typically, a FL course consists of multiple rounds of training. Figure 5.2 shows a basic round of an FL course and its actors implemented in FederatedScope. According to Xie et al. [102] this includes four major steps:

1. Global model broadcast: the server broadcasts the global model to all clients in the federation
2. Local training: Once clients have received the global model from the server, they start locally training the model with their private data
3. Client selection: The server chooses a percentage of clients to share their updated models for the global training round. This selection can be done randomly or a specified client selection strategy can be used.
4. Return updates: The server chooses a percentage of clients to share their updated models for the global training round. This selection can be done randomly or a specified client selection strategy can be used. The selected clients then share the locally trained model with the server.
5. Aggregation: The server performs federated aggregation on the received model updates with the help of an aggregator.

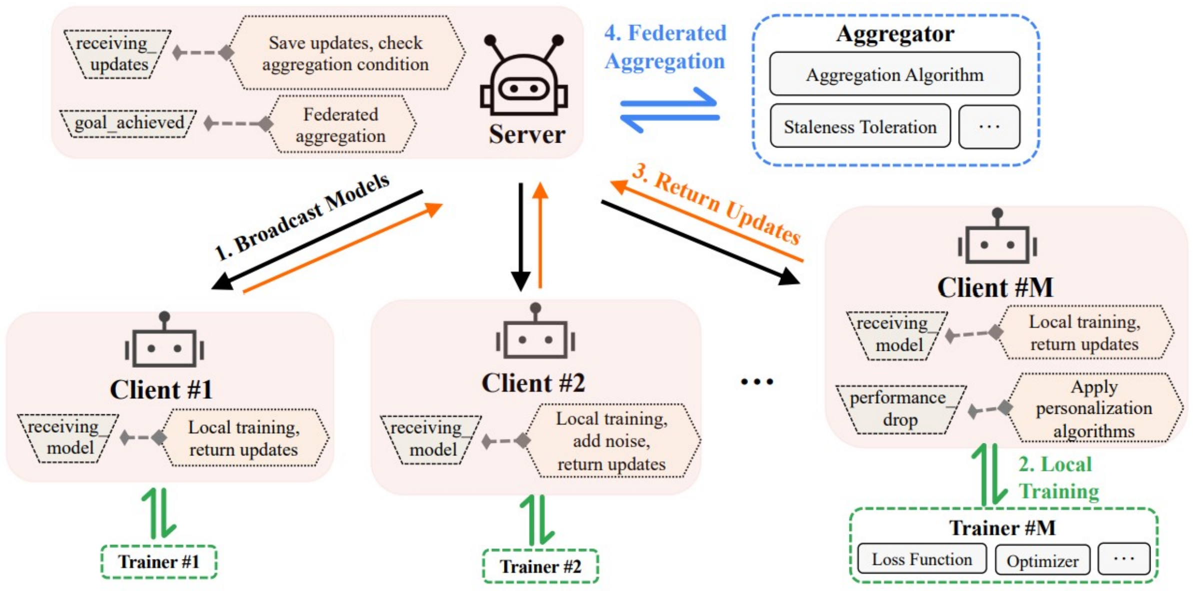


Figure 5.2: Overview of an FL round implemented with FederatedScope from [102].

5.1.2 CodeCarbon

CodeCarbon [90] is a lightweight software package that can be integrated into a Python codebase. It estimates the amount of carbon dioxide (CO_2) produced by the cloud or personal computing resources used to execute the code. It measures the power supply of underlying hardware at frequent time intervals. According to the developers, the package supports the following hardware [90]:

- GPU: energy consumption is tracked using the pynvml library
- RAM: an average value of 3 Watts per 8GB RAM is used to estimate RAM power consumption
- CPU: On Windows or Mac, CodeCarbon tracks CPU power consumption using the Intel Power Gadget, and on Linux, it tracks Intel processor power consumption from Intel RAPL files. In the case that none of the tracking tools are available for a computing resource, CodeCarbon uses a fallback approach. It identifies the specific CPU being used and matches it with a comprehensive data source containing information on over 2000 Intel and AMD CPUs, along with their respective TDPs. If the CPU is not found in the data source, however, a predefined global constant is used. CodeCarbon makes an approximation by assuming that 50% of the TDP represents the average power consumption. This methodology allows for an estimation of power consumption in case no direct tracking capabilities are available.

With the described methodology, the net power used is the net power supply measured in kWh consumed during the computing time [90].

The net power consumption is then multiplied by the carbon intensity of the energy grid used by the device. The carbon intensity of the energy grid is determined by the location of the device and the average carbon intensity of the energy grid of the country the device is located in, determined by a database containing countries and their carbon intensity of the energy grid [90].

CodeCarbon also provides a dashboard with visualizations explaining the energy that has been consumed, and the emissions that have been produced and put that measurements into perspective by comparing the emissions produced to monthly average emissions from a household, a car, or a television. An example dashboard can be seen in Figure 5.3.

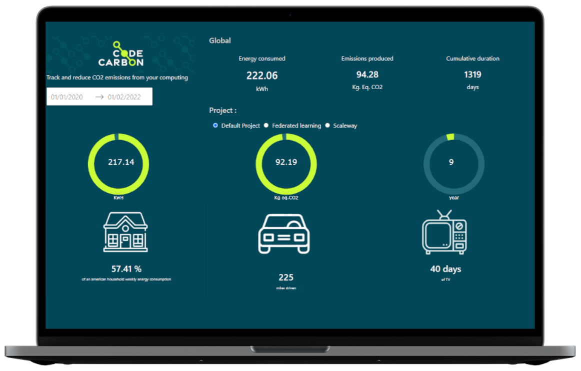


Figure 5.3: Exemplary CodeCarbon Emissions Dashboard from [90]

Even if the measurements taken by the package are not fully accurate, the package provides a fair estimation of emissions and helps to raise awareness of CO_2eq emissions of machine learning for developers and businesses. In this work, CodeCarbon version 2.2.1 is used.

5.1.3 FederatedTrust v.0.1.0

As already shortly introduced in Section 5.1.3 FederatedTrust is an algorithmic prototype developed by Sánchez et al. [14] to quantify the trustworthiness level of FL models according to the six pillars: privacy, robustness, fairness, explainability, accountability, and federation. Since FederatedTrust will be extended in this work with the sustainability pillar, this section gives a detailed introduction to FederatedTrust, its architecture, implementation, and metrics.

Design and Architecture

For the algorithm design, FederatedTrust considers the following four different input sources to compute the trustworthiness level of FL systems according to Sánchez et al. [14]. i) FL Model: contains information on the collaboratively trained model in the FL course. ii) FL framework configuration file: contains configuration parameters of FederatedScope needed to train and evaluate the FL model. This configuration includes the number of training rounds, the number of clients, the client selection rate, the dataset split, the client selection mechanisms, the model hyperparameters, the loss function, the optimizer, the model architecture, the aggregation algorithm, and more. iii) FactSheet: contains relevant information to calculate the trustworthiness score for all the pillars. The factsheet is filled through the FL course with information coming from the others sourced through FederatedTrust's TrustmetricManager. vi) Statistics: contains the statistical information extracted from the training dataset of each client such as client's class balance, client's test performance loss, and client's test accuracy.

Figure 5.4 visually represents the dynamic interactions among the various actors engaged in the computation process of determining the trustworthiness score computed by FederatedTrust integrated into an FL framework. The central server has the pivotal role of hosting four components: the aggregator, the FactSheet, the FL framework configuration, and the FederatedTrust algorithm. While the clients hold their local model, their private dataset, and client statistics [14].

The Aggregator takes on the responsibility of merging the parameters from the participating clients' models to construct the global FL model[14].

The FL framework configuration file holds detailed information about the FL system such as the number of participating clients, the model used and its architecture, the dataset used, the aggregation algorithm used, the number of global and local training rounds, and more [14].

The FactSheet, also residing on the central server, captures and documents essential information about the FL system and its actors that are later used to compute the notions and metrics for the trust FL score. Some information in the FactSheet is obtained from the configuration file and other information is coming from the FL course itself [14].

Furthermore, the central server holds the FederatedTrust algorithm, which takes on the computation of the final trust score for the FL system. By employing a comprehensive set of criteria and metrics, this algorithm evaluates the trustworthiness of the FL system and produces a final trust score that acts as an indicator of the FL system's trustworthiness level in terms of privacy, robustness, fairness, accountability, explainability, and federation [14].

Each client is responsible for local training and evaluation of the model with their own private data. Further, client statistics such as the performance of the model, the class distribution, and the entropy distribution are stored at the client's side [14].

Together, these components and actors form a cohesive FL framework supplied with FederatedTrust to compute the trustworthiness score of FL systems. To compute the trust score, during the pre-training phase, the relevant content of the FL framework configuration is sent to the FederatedTrust algorithm and copied to the FactSheet (steps one and two in Figure 5.4) and pre-training metrics are computed according to them. Then, the server shared the global model with all the clients in the federation, starting the local training process at each client (step three in Figure 5.4). At some point in time, the server chooses clients for the global aggregation, and the chosen clients sent back the model parameters of their locally trained models and the client's statistics to the server (step four in Figure 5.4). At that point, the model parameters are aggregated by the aggregator and FederatedTrust computes in-training metrics (step five in Figure 5.4). Until the number of global training rounds is reached, steps three, four, and five are repeated. Once reached this point, FederatedTrust computes the post-training metrics. In the end, the FederatedTrust algorithm reports a trust score per pillar and a global one of the federation. Additionally, a report is generated and stored [14]. A more detailed version of the steps is provided in the algorithmic pseudocode in Appendix 7.1.

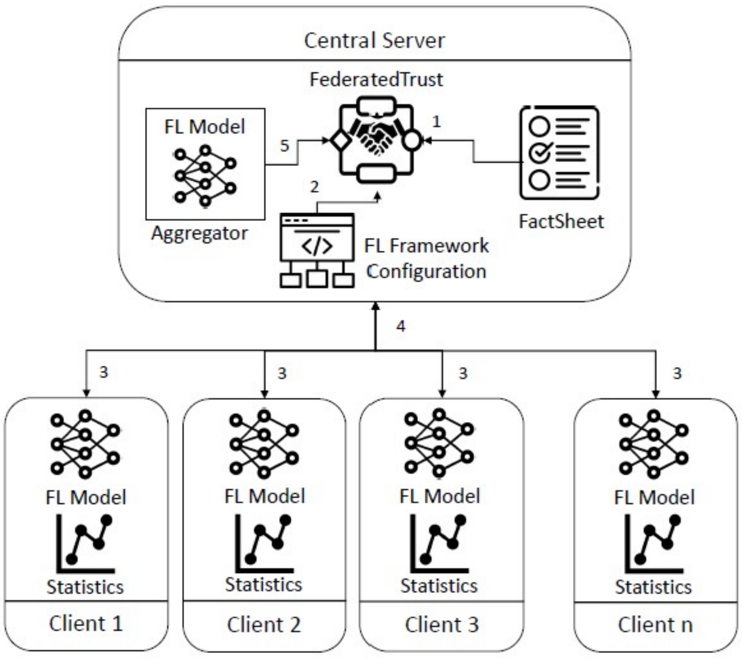


Figure 5.4: Overview of Different Actors of FederatedTrust integrated into a Framework from [14].

Metric and Metric Computation

The trust metrics that are implemented in FederatedScope have already been introduced and discussed in Section and are depicted in Table 3.3 but this subsection pays a closer look into the methodology used to compute this metrics and their normalization. As can be seen in Figure 5.5, the separate metrics are first going through metrics normalization functions, normalizing them to values between 0 and 1 [14]. How they are normalized, depends on the metric itself and the normalization function for each metric can be seen in Table A.2 in the Appendix.

After the normalization step, all metrics associated with a separate notion are passed to a metric aggregation function, which combines all metrics into a score for each notion. The notion score is a weighted average of all the metrics belonging to that notion. The weights can be adjusted in the evaluation metrics file of FederatedTrust. Out of the notion scores, the same aggregation is applied again to all the notions belonging to one trust pillar, resulting in a trust score for each of the six pillars. Then again, the same aggregation function is applied to all the six pillar scores finally resulting in a global trust score of the FL system [14].

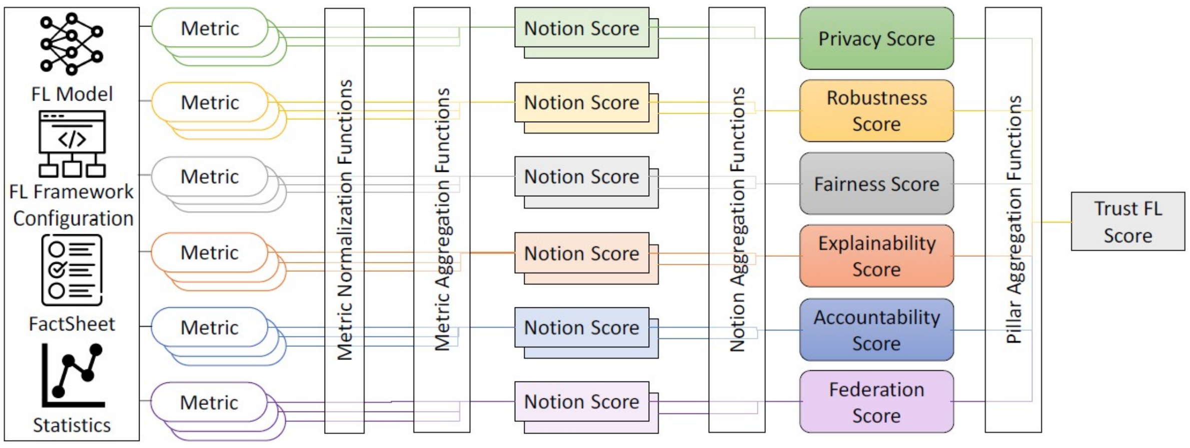


Figure 5.5: FederatedTrust Metric Calculation Process from [14].

Issues Experienced with FederatedTrust v.0.1.0

Whilst trying out FederatedTrust v.0.1.0 for this work, some problems with the package became present. Making the package as is run inside FederatedScope using the installation guidelines provided was not possible and the issues could only be resolved after inspecting another public repository of the developer that showed changes that were made to FederatedScope itself in order to make FederatedTrust run with it. These are only minor issues, but resolving these proved to be a time-consuming process. To spare this process to future users of FederatedScope v.0.1.0, this section provides a list of problems and steps that need to be taken in order to make FederatedTrust v.0.1.0 run:

- Missing files: the package is missing crucial files for metrics computation. It is missing a file for computing the clever metric as well as a file for computing the feature importance cv. These two files can be found under federatedscope/contrib/metrics/feature_importance_cv.py and federatedscope/contrib/metrics/clever.py in the repository available under this link: https://github.com/ningxie1991/FederatedScope. These files need to be downloaded and included under the same path in your local FederatedScope environment for FederatedTrust v.0.1.0 to run.

- Model zoo: Unfortunately, FederatedTrust v.0.1.0 does not work with the different models in the model zoo of FederatedScope. The definition of the model has been changed in order to work with some functions of FederatedTrust v.0.1.0 such as the computation of the SHAP value. To get FederatedTrust running, either the implementation of the SHAP value needs to be changed or the changed model definition of the developer has to be applied. The changed model definition can be found in the path federatedscope/cv/model/cnn.py in the following repository: https://github.com/ningxie1991/FederatedScope. One needs to replace the file federatedscope/cv/model/cnn.py in the local FederatedScope environment with the one that can be found under the link provided.

- Inclusion of code snippets: In order to make FederatedTrust v.0.1.0 work, code-snippets have to be integrated into FederatedScope. Some of them are described in the installation guidelines of FederatedScope v.0.1.0, but not all of them. In detail, the changes made to the files federatedscope/core/fed_runner.py, federatedscope/core/trainers/torch_trainer.py, federatedscope/core/workers/client.py and federatedscope/core/workers/server.py in the repository under the following link: https://github.com/ningxie1991/FederatedScope need to be applied to your local version of FederatedScope as well. An easier way is to download the named files and replace them in your local version of FederatedScope.

- Dependency issues: Once built FederatedTrust v.0.1.0 inside FederatedScope different dependency issues arise. Following versions of the external libraries needed work in the combination of FederatedScope and FederatedTrust:

```
torchvision == 0.15.2
adversarial-robustness-toolbox == 1.14.1
dotmap == 1.3.30
numpy==1.22.4
scipy==1.7.3
pandas == 2.0.1
hashids == 1.3.1
grpcio==1.55.0
grpcio-tools
protobuf == 3.20.3
pympler == 1.0.1
pyyaml==6.0
fvcore
iopath
wandb == 0.15.3
scikit-learn == 1.1.3
scipy == 1.7.3
shap == 0.41.0
```

```
tabulate == 0.9.0
tensorboard
tensorboardX
tensorflow == 2.12.0
torch == 2.0.1
pympler
```

- FederatedScope version: FederatedTrust v.0.1.0 works with FederatedScope v.0.2.0 once the above-mentioned adjustments have been taken, but not with the newest version v.0.3.0.

A simpler way to make FederatedTrust v.0.1.0 run within FederatedScope, is to clone the adjusted FederatedScope repository under this link:
https://github.com/ningxie1991/FederatedScope
and install FederatedTrust inside this environment. It is simpler since all the adjustments to files in FederatedScope are already done. However, the fixing of the dependency issues step described above still needs to be taken.

5.2 Design and Implementation of FederatedTrust v.0.2.0

This section explains the design and implementation of FederatedTrust v.0.2.0 which is an extension of FederatedTrust v.0.1.0 including the sustainability pillar and its metrics and notions. Subsection 5.2.1 defines the context and assumptions that were followed in designing and implementing the prototype. Subsection 5.2.2 defines requirements and constraints while Subsection 5.2.3 shows the architecture and Subsection 5.2.4 depicts the algorithmic pseudocode. Subsection 5.2.5 introduces the metrics and notions for the sustainability pillar and their definition. Subsection 5.2.6 explains the computation process for the metrics of the sustainability pillar. Subsection 5.2.7 handles the metric configuration file and the metric normalization process. Subsection 5.2.8 shows the process that is followed to compute the final trust score and Subsection 5.2.10 explains additional functionalities built into FederatedTrust v.0.2.0 that are not directly related to the computation of a final trustworthiness score. Lastly, Subsection 5.2.11 introduces the guidelines needed to follow to install and use FederatedTrust v.0.2.0.

5.2.1 Context, Assumptions

For the development of FederatedTrust v.0.2.0 included in the FederatedScope v.0.2.0 framework, the following contexts (C) and assumptions (A) are defined. The first context and the first assumption stayed the same as they were in FederatedTrust v.0.1.0 [79] but the second assumption is added to suit the new prototype.

- ”C-1: The use case that the algorithm is developed for is a client-server HFL model” [79].

- ”A-1: The central server is honest and maintained by a trusted system owner and thus does not interfere with the FL protocol maliciously” [79].
- A-2: Clients are honest but curious, meaning they do trustfully report their metrics and statistics without maliciously interfering with the FL protocol.

5.2.2 Requirements and Constraints

Similar to the first version of FederatedTrust [79, 14], this extended version of FederatedTrust assumes that the server is honest and maintained by a trusted system administrator, and thus the server does not maliciously interfere with the trust score computation process. The functional requirements (FR), non-functional requirements (NF), and privacy constraints (PC) for FederatedTrust v.0.2.0 are the same as they were in FederatedTrust v.0.1.0 and are the following:

”FR-1: Each of the seven trustworthy FL pillars must be represented in the algorithm, meaning that at least one metric from each pillar must be considered in the final score.

FR-2: The final trustworthiness score must be a combination of the trustworthiness scores from all notions and pillars.

NF-1: The algorithm should add minimal computation overhead and complexity to the server, participants, and FL model.

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

PC-1: The algorithm must not store any sensitive data from the FL model.

PC-2: The algorithm must not leak or share any sensitive data from clients, the server, and the FL model with third parties.

PC-3: The metrics calculations can occur at the client’s local devices, the central server, or collaboratively between both” [p. 10][14].

5.2.3 Architecture

Figure 5.6 depicts the architecture and interactions between FederatedScope and FederatedTrust v.0.2.0. It extends the original architecture of FederatedTrust v.0.1.0 [79] with the added components to obtain the metrics needed to compute the trust score of the sustainability pillar. The left side of Figure 5.6 shows the components coming from

FederatedTrust v.0.2.0 and the right side shows the components originally coming from FederatedScope. If FederatedTrust is mentioned, it relates to FederatedTrust v.0.2.0. The interactions between the different components involve the following steps:

1. Setup: Start federation by initiating the FedRunner from FederatedScope. The FedRunner takes the ConfigFile as input and initiates clients and the server as well as a TrustMetricManager from FederatedTrust. The TrustMetricManager takes the ConfigFile as input as well and begins populating the FactSheet with pre-training metrics such as the number of clients in the federation and the number of training rounds.

2. Model broadcast: The server broadcasts the global model to selected clients in the federation.

3. Local Training: The selected clients train their local models with their local private dataset. Additionally, clients are using functionalities of the CodeCarbon package (not shown in Figure 5.6) to obtain metrics relevant to the sustainability pillar computation such as the hardware and the carbon intensity of the energy grid used.

4. Report Emissions Metrics: Selected clients report metrics such as the hardware models and energy grid used to the TrustMetricManager which then stores it in the EmissionsFile.

5. Model Sharing: The selected clients then share their updated model parameters with the server.

6. Federated Aggregation: the Aggregator is used by the server and performs secure aggregation over the model updates received from selected clients.

7. Evaluation: After each training round, the clients perform model evaluation and call the MetricBuilder to perform metric calculations. The results of this evaluation are then written to the Eval Results File.

8. Next training round: Steps two to eight are repeated until all the training rounds are finished.

9. Propagate Evaluation Results: Once the final training round is finished and the FedRunner stops the collaborative training, The evaluation results get propagated to the FactSheet through the TrustMetricManager of FederatedTrust

10. Trust Score Computation: The TrustMetricManager calls the evaluation function to compute the overall trust score from the FactSheet and report including the trustworthiness scores get stored in the output directory of FederatedTrust.

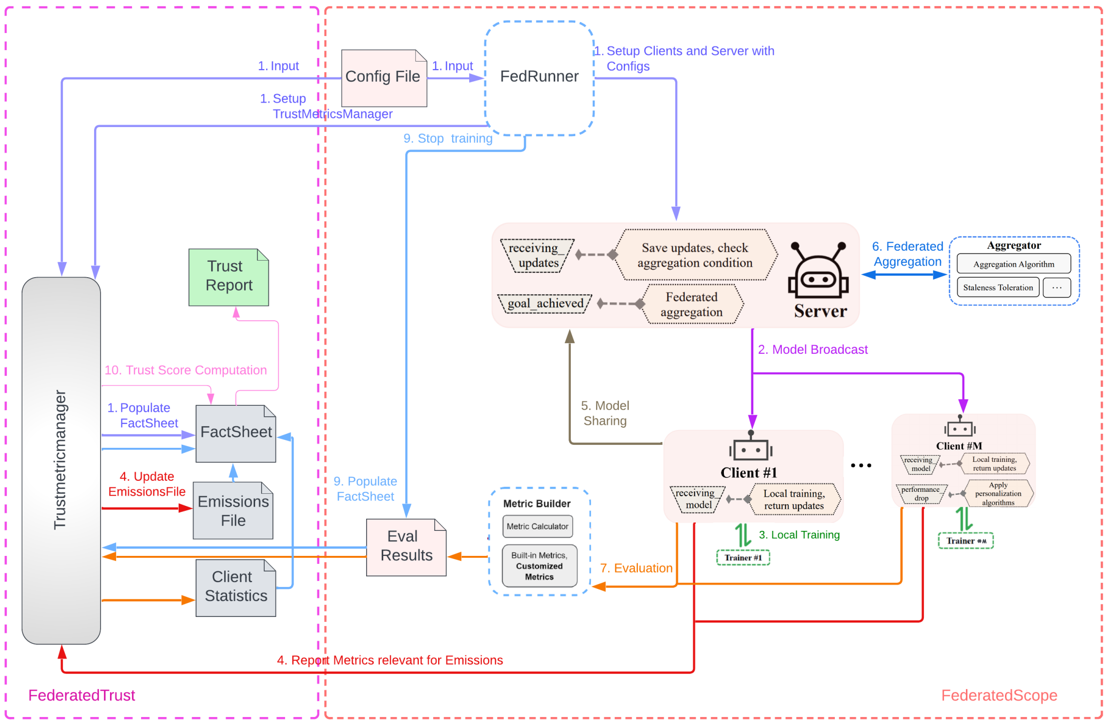


Figure 5.6: Interactions between FederatedTrust and FederatedScope, further development from [79] and [102].

5.2.4 Algorithmic Pseudocode

The execution of the training process for an FL model in FederatedScope, described in Subsection 5.6, along with the evaluation of its trustworthiness using FederatedTrust v.0.2.0, is shown in Algorithm 1. The algorithm is a further development of v.0.1.0 [14].

Algorithm 1 Training in FederatedScope with FederatedTrust v.0.2.0

Input: N clients, sampling size m, central server S, total number of iterations T, initial model $\overline{w}(0)$, setup configurations C, FederatedTrust metric manager ft

Output: Evaluation results, trustworthiness report, estimated carbon emissions

```
 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 the initial selection rate
 4: $S$ sends the model metadata to $ft$
 5: $S$ requests class distribution information from all clients $i \in [N]$
 6: $ft$ creates emissions file $ef$
 7: for each client $i \in [N]$ do
 8:     Client $i$ uses $ft$ function to calculate the sample size per class of local data
 9:     $ft$ creates or updates the class distribution map of hashed labels to sample size
10: end for
11: for $t = 0$ to $T$ do
12:     $S$ randomly samples $D(t) \subset [N]$ clients with size of $m$
13:     $S$ sends the hashed ids of the selected clients to $ft$
14:     $ft$ updates the client selection rate map
15:     $S$ broadcasts the current model $\overline{w}(t)$ to all clients $i \in D(t)$
16:     for each client $i \in D(t)$ do
17:         Client $i$ initializes an EmissionsTracker object from CodeCarbon and starts emissions
            tracking for training
18:         Client $i$ performs local training with $\overline{w}(t)$
19:         Client $i$ uses $ft$ function to stop emission tracking for training and to save results
20:         $ft$ updates $ef$
21:         Client $i$ sends new model updates $w(t+1)_i$ back to $S$
22:     end for
23:     $S$ initializes an EmissionsTracker object from CodeCarbon and starts emissions tracking
        for aggregation
24:     $S$ performs secure aggregation of all updates received into a new global model $\overline{w}(t+1)$
25:     $S$ uses $ft$ function to stop emissions tracking for aggregation and to save results
26:     $ft$ updates $ef$
27: end for
28: $S$ sends final global model $\overline{w}'$ to every client $i \in [N]$ for performance evaluation
29: for each client $i \in [N]$ do
30:     Client $i$ computes evaluation metrics with local test data and global model $\overline{w}'$
31:     Client $i$ sends the evaluation results back to S
32: end for
33: S aggregates the evaluation results and sends them to $ft$
34: $ft$ receives the evaluation results and populates the FactSheet with them
35: S asks $ft$ to evaluate the trustworthiness of the model
36: $ft$ computes the trustworthiness score and estimated emissions and generates a report JSON
    and print message
```

5.2.5 Metric Definitions

This Subsection contains the notions and metrics for the sustainability pillar in Table 5.1 as seen in the taxonomy 4.4. Each of the metrics has the following properties, similar to FederatedTrust v.0.1.0 [79]:

1. Metric: the name of the metric
2. Description: A short description/definition of the metric
3. Input: The document or data serving as input for metric calculation
4. Output: the raw output of the metric before the normalization step gets applied. The raw output can be in one of the following formats: - 0/1 - [m,n] - % - Integer - Real
   where [m,n] means that it is a range from m to n.
5. Dependency: dependent elements, packages, or libraries necessary for the computation

The other 20 metrics that have already been implemented in FederatedTrust v.0.1.0 can be seen in Figure A.2 in the Appendix 7.1 and more details about those metrics can be found in the original master thesis that implemented these metrics [79].

Table 5.1: Metrics for Sustainability Pillar.

Metric	Description	Input	Output	Dependency
Carbon Intensity of Energy Source				
Avg, carbon intensity of clients	The average carbon intensity of energy grid used by clients	location of clients (IP)	Float [20,795]	IP address of clients, codecarbon package incl. carbon intensity database
Carbon intensity server	The carbon intensity of energy grid used by the server	Location of server (IP)	Float [20,795]	IP address of server, codecarbon package incl. carbon intensity database
Hardware Efficiency				
Avg. hardware efficiency of clients	The average performance per watt (CPU or GPU Mark/ TDP) of CPUs and GPUs used by clients	CPU and GPU models of clients	Float [20,1447]	codecarbon package reads CPU and GPU model in use, hardware benchmark dataset from PassMark
Hardware efficiency of clients	The performance per watt (CPU or GPU Mark/ TDP) of CPUs and GPUs used by the server	CPU and GPU models of server	Float [20,1447]	codecarbon package reads CPU and GPU model in use, hardware benchmark dataset from PassMark
Federation Complexity				
Number of global training rounds	The number of global training rounds in the FL system	Config file	Integer	Is documented in the config file
Number of clients	The number of clients in the federation	Config file	Integer	Is documented in the config file
Client selection rate	% of clients selected in each training round to share their models	Config file	Float [0,1]	Is documented in the config file
Average number of local training rounds	The average number of local training rounds performed by clients withing one global training round	Config file	Integer	Is documented in the config file
Average dataset size	The average number of samples used by clients in one training round	Client Statistics	Integer	Is computed trough FederatedScope
Model size	Number of features/depth of decision tree/number of parameters in NN	Model	Integer	Model meta data

5.2.6 Raw Metric Computation

This Subsection explains how the computation of the raw metrics composing the notions of the sustainability pillar works. These raw metrics are afterward normalized with different normalization functions according to the metric configuration as described in Subsection 5.2.7. FederatedTrust v.0.2.0. still contains all the six other pillars that have already been implemented in FederatedTrust v.0.1.0. But since they were already there, they are not explicitly explained in this Subsection. For more details on metrics and notions composing all the pillars except the sustainability pillar please refer to [79].

Carbon Intensity of Energy Source

The carbon intensity of the energy source notion is compromised of two metrics: the average carbon intensity of the energy source used by all the clients in the federation and the carbon intensity of the energy source utilized by the server.

To calculate the average carbon intensity of the energy source used by clients metric, the EmissionsTracker functionality of the CodeCarbon package introduced in Subsection 5.1.2 is used. CodeCarbon obtains information about the location of the client and then matches this location with a database containing the average carbon intensity of the energy grid of countries. In each global training round, all the clients report their carbon intensity of the energy grid obtained by the CodeCarbon functionalities to the TrustMetricManager of FederatedTrust v.0.2.0, which then stores this information in a file. After the last global training round is completed, FederatedTrust computes the average of all the clients that reported energy grids in all training rounds.

A similar procedure happens on the server side to report the carbon intensity of the energy grid. However, since the server does not perform any training, the reporting happens when the server performs aggregation on the model parameters. The server also uses the CodeCarbon package to obtain the carbon intensity of its energy grid and reports it to the TrustMetricManager in each global round. Most of the time, the server does not change its location, thus the raw metric is equal to the carbon intensity of the energy grid the server is located in. If the server would change its location during the federated training of a model, FederatedTrust v.0.2.0 would still be able to capture it, since the metric is reported to the TrustMetricManager in every round, and after the last training round the average is computed.

Due to the implementation of the metrics included in the carbon intensity of energy source notion, FederatedTrust v.0.2.0 can precisely capture the metrics even if clients or server change their locations through the course of the federation or if clients are located in different countries of the world. Additionally, the CodeCarbon package is used and updated frequently, ensuring that the carbon intensity values of energy grids from countries remain up-to-date, ensuring that FederatedTrust v.0.2.0 does not become outdated quickly.

Hardware Efficiency

The hardware efficiency notion of the sustainability pillar is compromised by two notions: the average hardware efficiency of the clients and the hardware efficiency of the server.

Similarly to the carbon intensity of the energy source metrics, the hardware efficiency metrics rely on functions of the CodeCarbon package. CodeCarbon obtains the hardware that is installed in the clients and the server and during training on the client's side and aggregation at the server side also detects which part of the hardware, meaning if CPU or GPU, is used. This information is then sent from the worker to the TrustMetricManager and stored in each global round. After the last training round is finished, FederatedTrust uses a downloaded database in the form of a CSV file to match the CPU and GPU models with their power performance measurements. The CSV files containing the power performance measurements of CPUs [100] and GPUs [101] are created by PassMark, a widely used hardware benchmarking software, and are for this work downloaded from Kaggle and stored in FederatedTrust. After the CPU and GPU models are matched with their according power performance metrics, for all the clients the average of the power performance metrics of the used hardware is computed. Also for the server, the average is computed. But since the server mostly does not change its hardware, the average is the same as taking the power performance of one round.

With this implementation, FederatedTrust v.0.2.0 is able to accurately compute the hardware efficiency of the hardware that is used by clients and servers. It is also able to differentiate between hardware that is installed in the worker's devices and hardware that is actually used and only takes hardware that is used into account. Since the database that is used is a static CSV file, the database does not contain CPUs and GPUs that are developed in the future and thus needs to be updated. An improvement would be to build a crawler that directly gets the power performance values from the Passmarks website, such that the values are always up-to-date.

Federation Complexity

The federation complexity notion of the sustainability pillar is compromised of the number of training rounds, the number of clients in the federation, the client selection rate, the average number of local training rounds, the average dataset size, and the model size. All of these metrics are computed by the FederatedScope framework itself and are directly taken from it to further apply normalization functions by FederatedTrust.

5.2.7 Metric Configuration and Normalization

The metric configuration file serves as a centralized repository for storing all the essential pillars, notions, and metrics that are taken into account for calculating the trustworthiness score in FederatedTrust. In Figure 5.7, the structured representation of metric objects stored within the metrics configuration file can be observed. This JSON file contains all seven pillars, encompassing the notions associated with each pillar. Under each notion, a comprehensive list of metrics is provided as well as a weight for that notion. Each metric comprises various key-value pairs, including an input field that specifies the source from which the value of the metric can be obtained. Additionally, a field_path is included, which details the specific path within the source file where the value is stored.

By organizing this information within the metrics configuration file, FederatedTrust knows which metrics, notions and pillars need to be computed and considered during the trustworthiness evaluation process. For extending FederatedTrust with pillars, notions, or metrics in the future, this configuration file can be used.

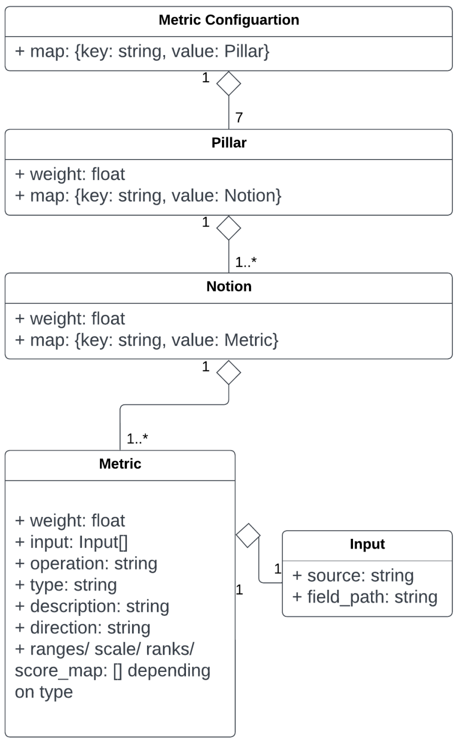


Figure 5.7: FederatedTrust v.0.2.0 Metric Configuration Design further Development from [79].

An important aspect to discuss in the configuration file is the type field. This field specifies the kind of normalization function that needs to be applied to the value of this metric in order to normalize it to a score between zero and one to later include it into the trust score of the notion and later the trust score of the pillar and ultimately the trust score of the federation. Currently, there are six different normalization functions implemented in FederatedTrust, but new functions can easily be implemented. The six functions include:

1. true_score: This type of metric directly reflects the output score based on the input. It is a straightforward mapping from input to output and thus also does not need a normalization function [79].

2. ranges: For this type of metric a specific range of values needs to be defined. The output score of this metric depends on which range the input value falls into. This allows considering different levels within a range, for example, models having one million, ten million, hundred million, or one billion parameters [79].

3. score_mapping: In this type of metric, the input value is mapped to an output value depending on a defined score map that must be provided in the score_map field [79].

4. score_ranking: With this type, we assign a ranking to the input value, and that ranking becomes the output score. It's more about the relative position or order of the input values, rather than their specific numerical values [79].

5. property_check: metrics of this type focus on verifying the presence or absence of specific properties in the input value. The output score is based on whether the property is present or not, enabling validation of essential characteristics [79].

6. scaled_score: metrics of this type lie in between a specified scale, for example between 20 and 120. This scale is then transformed to map the score between a scale of zero and one.

The first five metric types and their according normalization functions have already been implemented in FederatedTrust v.0.1.0 and their exact implementation and examples can be viewed at [79]. The Listing 5.1 shows the implementation of the new normalization function that is used for the newly introduced scaled_score metrics necessary for the sustainability pillars of FederatedTrust v.0.2.0.

```python
def get_scaled_score(value, scale:list, direction:str):
"""Maps a score of a specific scale into the scale between zero and one
    :param value: int or float: the raw value of the metric
    :param scale: list containing the minimum and maximum value the value can fall in between
    :param direction: asc means the higher the range the higher the score, desc means otherwise
    :return: normalized score of [0, 1]
"""
score = 0
low, high = 0, 1
try:
    value_min, value_max = scale[0], scale[1]
except Exception as e:
    logger.warning("Score minimum or score maximum is missing. The minimum has been set to 0 and the maximum to 1")
    value_min, value_max = 0,1
else:
    if value >= value_max:
        score = 1
    elif value <= value_min:
        score = 0
    else:
        diff = value_max - value_min
        diffScale = high - low
        score =  ((float(value) - value_min) * (float(diffScale) / diff) + low)
        if direction == 'desc':
            score = high - score
    return score
```

Code Listing 5.1: Code of Scaled Score Normalization Function.

The described scaled_score metric is used in the sustainability pillar of FederatedTrust v.0.2.0 for all the metrics belonging to the carbon intensity of the energy source pillar as well as all the metrics belonging to the hardware efficiency notion. Further, it is used for the client selection rate metric in the federation complexity notion. All the other metrics in the federation complexity pillar use score type number two ranges. The Listing 5.2 exemplary shows the part of the configuration file that represents the carbon intensity of the energy source notion of the sustainability pillar. Of course, the necessary fields for the other two notions of the sustainability pillar have also been added to the metric configurations file.

```
  "sustainability": {
    "energy_source": {
      "weight": 0.5,
      "metrics": {
        "carbon_intensity_clients": {
          "inputs": [
            {
              "source": "factsheet",
              "field_path": "sustainability/avg_carbon_intensity_clients
  "
            }
          ],
          "operation": "get_value",
          "type": "scaled_score",
          "direction": "desc",
          "scale": [20,795],
          "description": "Carbon intensity of energy grid used by
  clients",
          "weight": 0.5
        },
        "carbon_intensity_server": {
          "inputs": [
            {
              "source": "factsheet",
              "field_path": "sustainability/avg_carbon_intensity_server"
            }
          ],
          "operation": "get_value",
          "type": "scaled_score",
          "direction": "desc",
          "scale": [20,795],
          "description": "Carbon intensity of energy grid used by server
  ",
          "weight": 0.5
        }
      }
    }
```

Code Listing 5.2: Exemplary Part of Configuration File

5.2.8 Trust Score Computation

The procedure to calculate the overall trust score of the FL system in FederatedTrust v.0.2.0 is depicted in Figure 5.8. The metrics that are needed to compute the notion scores are obtained before, during, or after the training of the FL model, and are coming from different sources such as the FL model itself, the framework configuration file, the client statistics file, or the emissions file depending on the metric.

For the sustainability pillar of FederatedTrust v.0.2.0, the metrics are obtained from the FL model, framework configuration file, and the emissions file as discussed in Subsection 5.2.6. More information on the metrics of the other pillars and their inputs and computations can be found at [79].

In the first step, the metrics necessary to calculate the trust score are gathered and stored in a JSON file called FactSheet. The TrustMetricManager from FederatedTrust v.0.2.0 then collects these metrics from the FactSheet and applies the according normalization function to each metric to get a score between zero and one. Which normalization function has to be applied to which metric, is defined in the type field of each metric in the configuration file as discussed in Subsection 5.2.7.

After applying the normalization function, the metric aggregation function is called which aggregates all metrics belonging to a notion with a weighted average, resulting in trust scores per notion.

Then, the notion aggregation function is applied to aggregate all the notions belonging to a pillar with a weighted average to obtain a trust score per pillar.

To reach a final trust score, the pillar aggregation function is applied to aggregate all the notion trust scores and combine them to a final FL trust score again using a weighted average.

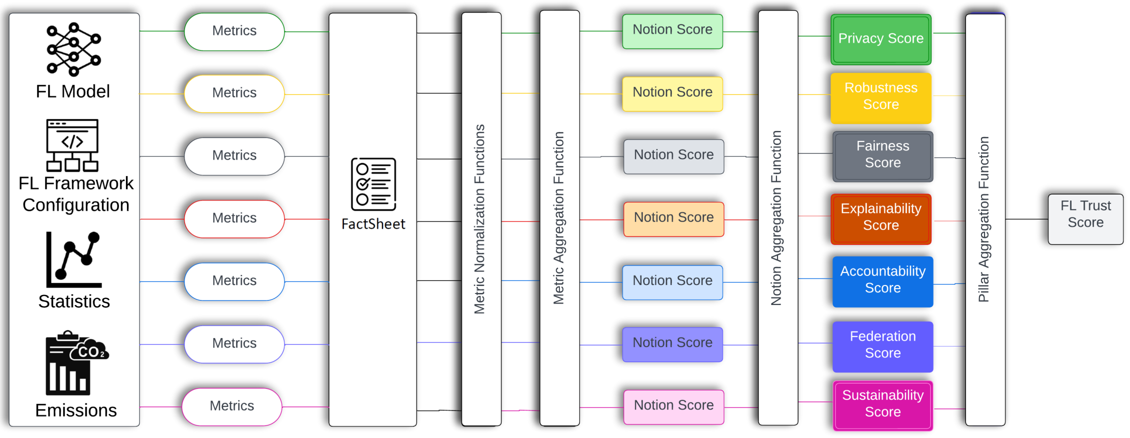


Figure 5.8: FederatedTrust v.0.2.0 Trust Score Computation Process, further development from [14].

5.2.9 Parametrization

Since running complex federations with lots of clients and a high number of training rounds just to compute the trust score of that configuration is time- and energy-consuming, FederatedTrust v.0.2.0 provides the possibility to parametrize configurations of the federation in order to compute the trust score. With this, users can compute the trust score a federation would get if it would have been run with other configurations e.g. 10e6 training rounds instead of 10 training rounds. To parametrize FederatedTrust v.0.2.0 one needs to store the metrics that want to be parametrized in the factsheet template file (FederatedTrust/configs/factsheet_template.json). An example factSheet template file where only the number of training rounds have been parametrized and the rest of the metrics have not been parametrized can be viewed in the Listing 5.3. The listing shows all the raw metrics that are gathered in the factsheet of FederatedTrust v.0.2.0 in order to compute the trust score. The parametrization of the number of training rounds in this file does not lead the federation to run 10e6 training rounds, instead, the number of training rounds that are specified in the configuration file are run. But the trust score is computed as if 10e6 training rounds would have been run. This is a useful feature if users want to play around with different configurations of federations and look at their resulting trust score without indeed running federations with these configurations.

```json
{
  "project": {
    "overview": "",
    "purpose": "",
    "background": ""
  },
  "data": {
    "provenance": "",
    "preprocessing": "",
    "avg_entropy": ""
  },
  "participants": {
    "client_num": "",
    "sample_client_rate": "",
    "client_selector": "",
    "avg_dataset_size": ""
  },
  "configuration": {
    "optimization_algorithm": "",
    "training_model": "",
    "personalization": "",
    "differential_privacy": "",
    "dp_epsilon": "",
    "trainable_param_num": "",
    "total_round_num": 10e6,
    "learning_rate": "",
    "local_update_steps": ""
  },
  "performance": {
    "test_loss_avg": "",
    "test_acc_avg": "",
    "test_feature_importance_cv": "",
    "test_clever": ""
  },
  "fairness": {
    "test_acc_cv": "",
    "selection_cv": "",
    "class_imbalance": ""
  },
  "system": {
    "avg_time_minutes": "",
    "avg_model_size": "",
    "avg_upload_bytes": "",
    "avg_download_bytes": "",
    "total_upload_bytes": "",
    "total_download_bytes": ""
  },

  "sustainability": {
    "avg_carbon_intensity_server": "",
    "avg_carbon_intensity_clients": "",
    "avg_power_performance_clients": "",
    "avg_power_performance_server": "",
    "emissions_training": "",
    "emissions_aggregation": "",
    "emissions_communication_uplink": "",
    "emissions_communication_downlink": ""

  }
}
```

Code Listing 5.3: Exemplary Part of Configuration File

5.2.10 Additional Functionalities

In addition to calculating the trust score of a federation, FederatedTrust v.0.2.0 provides another useful functionality: estimating the emissions that are produced by the federation. This estimation relies on the EmissionsTracker of the CodeCarbon package introduced in Subsection 5.1.2. The estimated emissions in g of CO_2eq are printed at the end of the training process in the console by FederatedTrust v.0.2.0. The emissions produced are obtained and estimated by CodeCarbon in the training stage at the clients' side and the aggregation stage at the server side. In addition, FederatedTrust also estimates the emissions produced for communicating the model parameters up (from the clients to the server) and down (from the server to the clients). For this estimation, the number of bytes that are communicated metric, which is computed by the FederatedScope framework itself, is taken and multiplied by 2.24e-10 kWh/byte, resulting in an estimated energy consumption to transfer the number of bytes. This constant is an estimation published by the Shift Project in its one-byte report [103]. The energy consumption is then multiplied by the carbon intensity of the server to obtain the emissions produced by downlink communication and multiplied by the average carbon intensity of the clients to obtain the uplink communication. The estimated emissions for training, aggregation, uplink communication, and downlink communication are summed and printed in the console. It is important to note, that the emissions shown in the console are only an estimate, due to using constants for the communication part. Of course, the energy consumption of transferring data over the internet depends on many factors such as the distance and the internet protocol used.

5.2.11 Installation Guidelines

In order to simplify the time-consuming installation process of FederatedTrust v.0.1.0, this new version FederatedTrust v.0.2.0 is directly integrated into FederatedScope. Thanks to this approach, the installation is as simple as cloning the repository from GitHub and installing the dependencies through the setup.py file. The dependency issues described in Subsection 5.1.3 have been resolved in this version.

The installation process includes:

1. Cloning the GitHub repository

```
git clone https://github.com/lzumta/FederatedScope.git
```

2. Changing the directory to FederatedScope

```
cd FederatedScope
```

2. Installing Dependencies into a virtual environment

```
# Editable mode
pip install -e .
```

```
# Or (developers for dev mode)
pip install -e .[dev]
pre-commit install
```

After the installation, the example experiment can be executed with these steps:

1. Changing the directory to federatedTrust

```
cd federatedTrust
```

2. Executing example federation

```
python ../federatedScope/main.py --cfg configs/example_config.yaml
```

To apply your own configurations, create a *config.yaml* file and pass it to FederatedScope.

To apply parametrization, change the values of the metrics you wish to parametrize in the *federatedTrust/configs/factsheet_template.json* file.

More detailed instructions for configuration and parametrization are provided in the ReadMe of the repository.

Chapter 6

Evaluation, Results, and Discussion

This chapter provides a comprehensive assessment of the sustainability pillar as well as the algorithmic prototype including all seven pillars. With seven different evaluation scenarios, the capabilities of the sustainability pillar, the algorithm including all pillars, and their limitations are explored.

Since the algorithm directly computes metrics for the sustainability pillar through running the federation, such as the hardware that is used by clients, setting up different evaluation scenarios would mean running the federation on different devices. Further, to get different values for the carbon intensity of the energy grid used, the federation would need to be run in different countries. Additionally, running complex federations with hundreds of training rounds and clients would be computationally expensive and take tremendous amounts of time. To simplify these aspects, parametrization 5.2.9 is used in some of the evaluation scenarios.

First, one evaluation scenario (Section 6.1) is run without parameterizing metrics and thus using FederatedTrust v.0.2.0 as intended. Then, the same scenario is run with parametrized metrics, confirming that the algorithm produces consistent results, no matter if parameters have been used or the values are directly computed from the federation and thus validating the functionality of the algorithm.

Second, four different parametrized evaluation scenarios are run (Section 6.2), analyzing the behavior of FederatedTrust v.0.2.0 focusing on the sustainability pillar. Various aspects are analyzed, such as different complexities of the federation, different magnitudes of the carbon intensity of the energy grid used by the clients and the server, and different hardware efficiencies of CPUs used by clients and the server by parametrizing FederatedTrust v.0.2.0.

Third, two evaluation scenarios are run (Section 6.3), analyzing the behavior of FederatedTrust v.0.2.0 as a whole including all the seven pillars.

Finally, the results of the experiments are discussed as well as the limitations of the algorithmic prototype implementation and design (Section 6.4).

6.1 Parametrized vs. Non-parametrized

This section evaluates the functionality of FederatedTrust v.0.2.0 by running two evaluation scenarios using the algorithm as intended, meaning that all the metrics are obtained and computed directly from the federation and its participants as introduced in Subsection 5.2.6 without applying any parametrization. Then, the exact same evaluation scenario is run with parametrization to evaluate the behavior of the algorithm with and without parametrization and ensure similar outputs. Parametrization means manually setting values to metrics instead of letting them be obtained through the federation as explained in Subsection 5.2.9.

6.1.1 Scenario Zero

Scenario Description

In scenario zero, a small federation is run with ten training rounds, ten clients, a client sample rate of one, and one local training round, using the FEMNIST dataset with a size of 100 samples at each client, and a small DNN model with two layers and a total of 5'730'000 parameters. The server as well as all the ten clients are located in Switzerland, which has a low carbon intensity of the energy grid of 32.87. The server as well as all the ten clients use an Intel(R) Core(TM) i7-8650U CPU @ 1.90GHz which has a power performance of 423.89.

This scenario is once run without parametrization, meaning that FederatedTrust v.0.2.0 obtains the metrics directly. The values, metrics, and scores of this run are depicted in Table 6.1. Then, this scenario is once run with parametrization of the metrics, and the results are shown in Table 6.2.

Results

As the comparison of the non-parametrized run of FederatedTrust v.0.2.0 in Table 6.1 and the parametrized run in Table 6.2 shows, the metrics that the algorithm obtains to the hardware model of the clients and the server, as well as the metrics of the carbon intensities of the energy grid used are correct as the values are the same in both Experiments. Further, the scores that the algorithm outputs are the same in the non-parametrized and parametrized version. The only difference is the source of the values, as one experiment is parametrized (Table 6.2) and one is not parametrized (Table 6.2) as shown in the source column of the Tables. Out of this, it can be concluded, that the algorithm obtains and computes the metrics for the sustainability pillar correctly from the federation and the third-party package CodeCarbon.

Table 6.1: Evaluation Scenario Zero not Parametrized

Sustainability Pillar			0.77
Metric	Value	Source	Score
Carbon intensity of energy source (weight 0.5)			0.98
Avg. carbon intensity of energy grid clients	32.87g CO_2eq / kWh	Computed	0.98
Carbon intensity of energy grid server	32.97g CO_2eq / kWh	Computed	0.98
Hardware efficiency (weight 0.25)			0.28
Avg. hardware efficiency clients	423.89	Computed	0.28
hardware efficiency server	423.89	Computed	0.28
Federation complexity (weight 0.25)			0.83
Number of training rounds	10	ConfigFile	1
Number of clients	10	ConfigFile	1
Client selection rate	1	ConfigFile	0.0
Average number of local training rounds	1	ConfigFile	1
Average dataset size	100	Computed	1
Model size	5’730’000	Computed	1

Table 6.2: Evaluation Scenario Zero Parametrized

Sustainability Pillar			0.77
Metric	Value	Source	Score
Carbon intensity of energy source (weight 0.5)			0.98
Avg. carbon intensity of energy grid clients	32.87g CO_2eq / kWh	Parametrized in FactSheet	0.98
Carbon intensity of energy grid server	32.97g CO_2eq / kWh	Parametrized in FactSheet	0.98
Hardware efficiency (weight 0.25)			0.28
Avg. hardware efficiency clients	423.89	Parametrized in FactSheet	0.28
hardware efficiency server	423.89	Parametrized in FactSheet	0.28
Federation complexity (weight 0.25)			0.83
Number of training rounds	10	Parametrized in FactSheet	1
Number of clients	10	Parametrized in FactSheet	1
Client selection rate	1	Parametrized in FactSheet	0.0
Average number of local training rounds	1	Parametrized in FactSheet	1
Average dataset size	100	Parametrized in FactSheet	1
Model size	5’730’000	Parametrized in FactSheet	1

6.2 Sustainability Pillar Evaluation of FederatedTrust v.0.2.0

From the definition of the sustainability pillar of trustworthy FL, there exist different combinations of FL systems having different carbon-intensive energy grids, different efficiency of hardware, and different federation complexities resulting in different magnitudes of CO_2eq emissions. Out of that, there exist four edge cases that can be seen in Figure 6.1. On the bottom left, the best-case scenario is located. The best case scenario is a simple federation using efficient hardware and thus having low energy consumption and additionally using a low carbon-intensive energy grid. The combination of low energy consumption and low carbon intensity of the energy grid results in low carbon emissions and should thus receive a high score in the sustainability pillar of trustworthy FL. On the contrary, having a high energy consumption due to using not efficient hardware and a complex federation in combination with using a high-intensity energy grid leads to high carbon emissions and should thus result in a low sustainability score. On the top left of Figure 6.1, a middle-case scenario with low energy consumption but a carbon-intensive energy grid is situated. On the contrary, on the bottom right a middle-case scenario with high energy consumption but a not carbon-intensive energy grid is situated. Both of these scenarios lead to medium CO_2eq emissions and should thus get a medium score in the sustainability pillar. The following Subsections provide an evaluation scenario for each of the four cases described above to research if the resulting sustainability score adheres to the expectations.

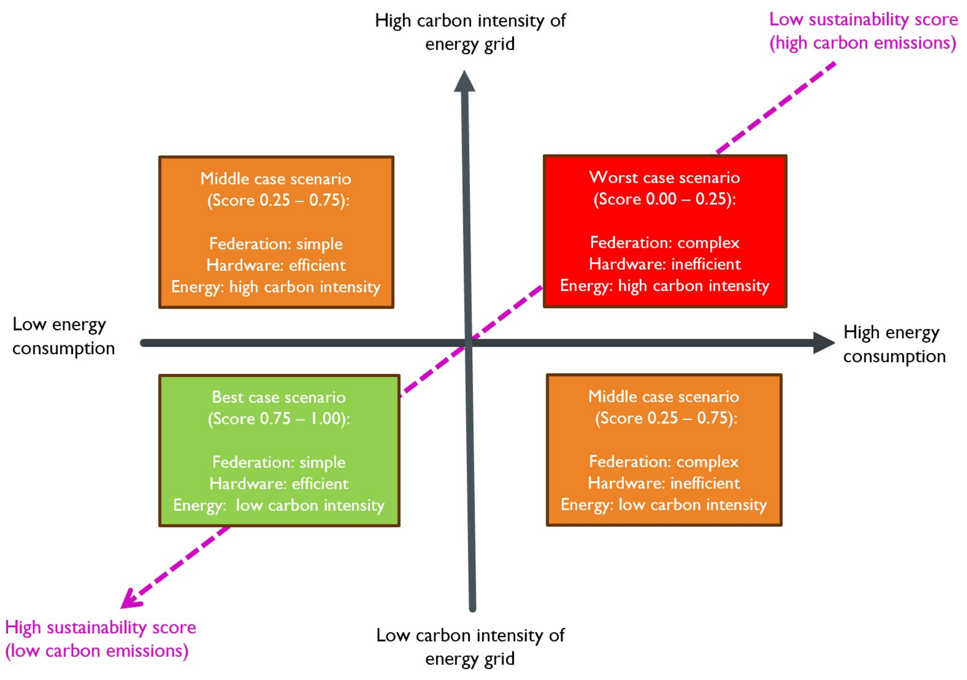

High carbon intensity of energy grid
Low sustainability score (high carbon emissions)
Middle case scenario (Score 0.25 – 0.75): Federation: simple Hardware: efficient Energy: high carbon intensity
Worst case scenario (Score 0.00 – 0.25): Federation: complex Hardware: inefficient Energy: high carbon intensity
Low energy consumption
High energy consumption
Best case scenario (Score 0.75 – 1.00): Federation: simple Hardware: efficient Energy: low carbon intensity
Middle case scenario (Score 0.25 – 0.75): Federation: complex Hardware: inefficient Energy: low carbon intensity
High sustainability score (low carbon emissions)
Low carbon intensity of energy grid

Figure 6.1: The Sustainability Pillar Matrix

6.2.1 Scenario One: Best Case Scenario

Scenario Description

Scenario one represents the best-case scenario with low CO_2eq emission. In this scenario, the Intel Core i7-1250U CPU is used by the server as well as all five clients, which is efficient with a power performance of 1447, the highest measured by PassMark so far. Further, the federation complexity is low, with a small number of clients, global training rounds, local training rounds, and a small client selection rate, dataset size, and model size. Additionally, the clients and the server are located in Albania with one of the lowest carbon-intensive energy grids. The parameters and evaluation scores of the best-case scenario can be seen in Table 6.3.

Results

As Table 6.3 shows, the carbon intensity of the energy source of the server is scored as one, since the server is located in Albania with a low carbon intensity energy grid of 23 kg of CO_2eq per kWh. Similarly, the average carbon intensity of the energy grid of the five clients is scored as one, since all of them are also located in Albania. These two metrics combined by the metric aggregation function result in a score of one for the carbon intensity of the energy sources used notion.

The hardware efficiency of the server is scored as one since the server is using an Intel Core i7-1250U CPU having a high power performance of 1447. Similarly, all five clients are using this CPU and thus also the average hardware efficiency of clients is scored as one. These two metrics combined by the metric aggregation function result in a score of one for the hardware efficiency notion.

In terms of federation complexity, the number of training rounds is low with only ten training rounds, and thus scored by the algorithm as one. Similarly, the federation only contains five clients, which is low and thus scored as one by the normalization function of the algorithm. The client selection rate of 0.2 is scored as 0.89 since it is low but the lowest set by the normalization function is a client selection rate of 0.1. The average number of local training rounds is only one, the lowest possible, and thus scored as one. The average dataset size on the client's side is only 100 samples per global training round, which is low and is thus scored as one by the algorithm. Lastly, the federation in evaluation scenario one uses a small DNN model with only 98'000 parameters, which is scored as one by the algorithm. All these metrics together result in a high federation complexity score of 0.98 since the federation complexity is at the lower end.

Combining these three notions with the weighted average, the overall score for the sustainability pillar is one for scenario one which represents a best-case scenario in terms of the sustainability pillar with efficient hardware, a small federation, and a not carbon-intensive energy grid and thus low emissions.

Table 6.3: Evaluation Scenario One

Sustainability Pillar		1
Metric	Value	Score
Carbon intensity of energy source (weight 0.5)		1
Avg. carbon intensity of energy grid clients	23g CO_2eq / kWh	1
Carbon intensity of energy grid server	23g CO_2eq / kWh	1
Hardware efficiency (weight 0.25)		1
Avg. hardware efficiency clients	1447	1
hardware efficiency server	1447	1
Federation complexity (weight 0.25)		0.98
Number of training rounds	10	1
Number of clients	5	1
Client selection rate	0.2	0.89
Average number of local training rounds	1	1
Average dataset size	100	1
Model size	98'000	1

6.2.2 Scenario Two: Worst Case Scenario

Scenario Description

Scenario two represents a worst-case scenario with inefficient hardware and a highly complex federation resulting in high energy consumption and high carbon intensity of the electricity grid used and thus, high CO_2eq emission. In this scenario, an Intel Xeon W-2104 CPU having a low power performance of 51.67 is used by the server. All the 10e6 clients use an AMD FX-9590 CPU, which has a low power performance of 30.76. Thus, in total, the hardware used is not efficient. Further, the federation complexity is high, with 10e6 clients, 10e6 global training rounds, 90 local training rounds, 100% client selection rate, 1.1 * 10e5 dataset size, and 10e12 number of parameters in the DNN model. Additionally, the server is located in South Africa with one of the highest possible energy grids with 709g CO_2eq per kWh. Half of the clients are located in Kosovo, with a carbon-intensive energy grid of 769g of CO_2eq per kWh and the other half is located in Gambia with a carbon-intensive electricity grid of 700g of CO_2 eq per kWh, resulting in an average carbon intensity of the electricity grid used by clients of 734.5g of CO_2eq per kWh. The parameters and evaluation scores of scenario two can be seen in Table 6.4.

Results

As Table 6.4 shows, the carbon intensity of the energy source of the server is scored as 0.11, since the server is located in South Africa with a high carbon intensity energy grid of 709g of CO_2eq per kWh. Similarly, the average carbon intensity of the energy grid of the 10e6 clients is scored as 0.08, since 50% of them are located in Kosovo and the other half in Gambia, both having high carbon-intensive energy grids. These two metrics combined by the metric aggregation function result in a score of 0.09 for the carbon intensity of the energy sources used notion.

The hardware efficiency of the server is scored as 0.01 since the server is using an Intel Xeon W-2104 CPU having a low power performance of 51.67. All the clients are using a low-power performance CPU and thus also the average hardware efficiency of clients is scored as 0.01. These two metrics combined with the metric aggregation function result in a score of 0.01 for the hardware efficiency notion.

In terms of federation complexity, the number of training rounds is high with 10e6 training rounds and thus scored by the algorithm as 0.17. Similarly, the federation contains 10e6 clients, which is high and thus scored as 0.17 by the normalization function of the algorithm. The client selection rate of 1 is scored as 0.00 since it is the highest possible client selection rate and every client shares its parameters in every global training round. The average number of local training rounds is only 90 and scored by the normalization function as 0.11. The average dataset size at the client's side is 1.1 * 10e5 samples per global training round, which is high and is thus scored as 0.2 by the algorithm. Lastly, the federation in evaluation scenario two uses a big DNN model with only one billion parameters, which is scored as 0.14 by the algorithm. All these metrics together result in a high federation complexity score of 0.13 since the federation is rather complex.

Combining these three notions with the weighted average, the overall score for the sustainability pillar is 0.09 for scenario two which represents a worst-case scenario in terms of the sustainability pillar using inefficient hardware and carbon-intensive electricity grids in combination with a complex federation.

Table 6.4: Evaluation Scenario Two

Sustainability Pillar		0.09
Metric	Value	Score
Carbon intensity of energy source (weight 0.5)		0.09
Avg. carbon intensity of energy grid clients	734.5g CO_2eq / kWh	0.08
Carbon intensity of energy grid server	709g CO_2eq / kWh	0.11
Hardware efficiency (weight 0.25)		0.01
Avg. hardware efficiency clients	30.76	0.01
hardware efficiency server	51.67	0.02
Federation complexity (weight 0.25)		0.13
Number of training rounds	10e6	0.17
Number of clients	10e6	0.17
Client selection rate	1	0.00
Average number of local training rounds	90	0.10
Average dataset size	1.1 * 10e5	0.2
Model size	10e12	0.14

6.2.3 Scenario Three: Middle Case Scenario

Scenario Description

Scenario three represents a middle case scenario with inefficient hardware and a complex federation resulting in high energy consumption but a low carbon intensity of electricity grid used and thus, medium CO_2eq emission. In this scenario, the Intel Core i7-6800K @ 3.40GHz CPU is used by the server, which has a power performance of 76.29. 40% of the clients use an Intel Xeon E5-4620 v3 @ 2.00GHz CPU, with a power performance of 100.24. 35% of clients use an Intel Xeon E5-4627 v2 @ 3.30GHz, which has a power performance of 71.69 and 25% use an Intel Xeon E5-2650 v2 @ 2.60GHz which has a power performance of 105.21. Further, the federation complexity is high, with 10e6 clients, 10e6 global training rounds, 90 local training rounds, 80% client selection rate, 1.1 * 10e5 dataset size, and 10e12 number of parameters in the DNN model. The server and the clients are located in Switzerland with an energy grid of 32g CO_2eq per kWh. The parameters and evaluation scores of scenario three can be seen in Table 6.5.

Results

As Table 6.5 shows, the carbon intensity of the energy source of the server is scored as 1, since the server is located in Switzerland with a low carbon intensity energy grid of 32g of CO_2eq per kWh. Similarly, the average carbon intensity of the energy grid of the 10e6 clients is scored as 1, since they are also located in Switzerland. These two metrics combined by the metric aggregation function result in a score of 1 for the carbon intensity of the energy sources used notion.

The hardware efficiency of the server is scored as 0.05 since the server is using Intel Core i7-6800K @ 3.40GHz CPU, which has a low power performance of 76.29. All the clients are using a low-power performance CPU and thus also the average hardware efficiency of clients is scored as 0.04. These two metrics combined with the metric aggregation function result in a score of 0.04 for the hardware efficiency notion.

In terms of federation complexity, the number of training rounds is high with 10e6 training rounds and thus scored by the algorithm as 0.17. Likewise, the federation contains 10e6 clients, which is high and thus scored 0.17 by the normalization function of the algorithm. The client selection rate of 0.8 is scored as 0.22. The average number of local training rounds is 90 and scored by the normalization function as 0.11. The average dataset size at the client's side is 1.1 * 10e5 samples per global training round, which is high and is thus scored as 0.2 by the algorithm. Lastly, the federation in evaluation scenario three uses a big DNN model with only one billion parameters, which is scored as 0.14 by the algorithm. All these metrics together result in a high federation complexity score of 0.17 since the federation is rather complex.

Combining these three notions with the weighted average, the overall score for the sustainability pillar is 0.55 for scenario three which represents a middle-case scenario with high energy consumption but low carbon intensity of the energy grid used.

Table 6.5: Evaluation Scenario Three

Sustainability Pillar		0.55
Metric	Value	Score
Carbon intensity of energy source (weight 0.5)		1
Avg. carbon intensity of energy grid clients	23g CO_2eq / kWh	1
Carbon intensity of energy grid server	32g CO_2eq / kWh	1
Hardware efficiency (weight 0.25)		0.04
Avg. hardware efficiency clients	91.49	0.05
hardware efficiency server	76.29	0.04
Federation complexity (weight 0.25)		0.17
Number of training rounds	10e6	0.17
Number of clients	10e6	0.17
Client selection rate	0.8	0.22
Average number of local training rounds	90	0.10
Average dataset size	1.1 * 10e5	0.2
Model size	10e12	0.14

6.2.4 Scenario Four: Middle Case Scenario

Scenario Description

Scenario four represents a middle case scenario with a simple federation and highly efficient hardware, thus low energy consumption but a highly carbon-intensive energy grid resulting in medium CO_2eq emission. In this scenario, the Intel Core i7-1250U CPU power performance of 1447 is used by the server. All eight clients use the Intel Core i5-1335U with a power performance of 1268. Further, the federation complexity is low, with a small number of clients, global training rounds, local training rounds, and a small client selection rate, dataset size, and model size. Moreover, the clients and server are located in South Africa with one of the most carbon-intensive energy grids with 709g CO_2eq per kWh. The parameters and evaluation scores of the middle case scenario four can be seen in Table 6.6.

Results

As Table 6.6 shows, the carbon intensity of the energy source of the server as well as of the clients is scored as 0.11 since the server and all the clients are located in South Africa using a carbon-intensive energy grid of 709 kg of CO_2eq per kWh. These two metrics combined by the metric aggregation function result in a score of 0.11 for the carbon intensity of the energy sources used notion.

The hardware efficiency of the server is scored as one since the server is using an Intel Core i7-1250U CPU having a high power performance of 1447. All eight clients are using the Intel Core i5-1335U CPU with a power performance of 1268 and this metric is scored by the normalization function as 0.87. These two metrics combined with the metric aggregation function result in a score of 0.94 for the hardware efficiency notion.

In terms of federation complexity, the number of training rounds is low with only ten training rounds, and thus scored by the algorithm as one. Alike, the federation only contains eight clients, which is low and thus scored as one by the normalization function of the algorithm. The client selection rate of 0.3 is scored as 0.77 since it is low but the lowest set by the normalization function is a client selection rate of 0.1. The average number of local training rounds is only one, the lowest possible, and thus scored as one. The average dataset size on the client's side is only 100 samples per global training round, which is low and is thus scored as one by the algorithm. Lastly, the federation in evaluation scenario four uses a small DNN model with only 99'300 parameters, which is scored as one by the algorithm. All these metrics together result in a high federation complexity score of 0.96 since the federation complexity is at the lower end.

Combining these three notions with the weighted average, the overall score for the sustainability pillar is 0.53 for scenario four which represents a middle-case scenario with efficient hardware and a small federation and thus low energy consumption, but a carbon-intensive energy grid and thus medium CO_2eq emissions.

Table 6.6: Evaluation Scenario Four

Sustainability Pillar		0.53
Metric	Value	Score
Carbon intensity of energy source (weight 0.5)		0.11
Avg. carbon intensity of energy grid clients	709g CO_2eq / kWh	0.11
Carbon intensity of energy grid server	709g CO_2eq / kWh	0.11
Hardware efficiency (weight 0.25)		0.94
Avg. hardware efficiency clients	1268	0.87
hardware efficiency server	1447	1
Federation complexity (weight 0.25)		0.96
Number of training rounds	10	1
Number of clients	8	1
Client selection rate	0.3	0.77
Average number of local training rounds	1	1
Average dataset size	100	1
Model size	99'300	1

6.3 Evaluation of FederatedTrust v.0.2.0

This section evaluates the prototype FederatedTrust v.0.2.0 including all seven pillars: privacy, fairness, robustness, federation, explainability, accountability, and sustainability with two different evaluation scenarios using different federation configurations. These evaluation scenarios are used to compare FederatedTrust v.0.2.0 and FederatedTrust v.0.1.0.

6.3.1 Scenario a

Scenario Description

The configurations for this evaluation scenario a can be viewed in Table 6.7. The federation uses one server and 10 clients, all located in Switzerland using an Intel Core i7-8650U CPU. The model is a 2-layer CNN and the dataset used is the FEMNIST dataset to classify handwritten digits. The dataset is split into 60% used for training, 20% for testing and 20% for evaluation. The federation runs for 10 global training rounds and clients perform only one local training round per global training round. In each global training round, 60% of clients are selected to share their model parameters with the server. Differential privacy is on with an *epsilon* of 10. The last column of Table 6.7 shows the source where these metrics have been set. In this experiment, all the values are configured in the ConfigFile or obtained through the course of the federation. No parametrization has been applied,

Table 6.7: Configurations FederatedTrust v.0.2.0 evaluation scenario a

Metric	Value	Source
Model	ConvNet2 (FederatedScope)	ConfigFile
Number of local training rounds	10	ConfigFile
Dataset	FEMNIST	ConfigFile
Data split	0.6/0.2/0.2 (train, test, validation)	ConfigFile
Batch size	50	ConfigFile
Loss	CrossEntropyLoss	ConfigFile
Consistent label distribution	False	ConfigFile
Number of clients	10	ConfigFile
Client Selection rate	60%	ConfigFile
Number of training rounds	10	ConfigFile
Client hardware	Intel(R) Core(TM) i7-8650U CPU @ 1.90GHz	Obtained
Server hardware	Intel(R) Core(TM) i7-8650U CPU @ 1.90GHz	Obtained
Client location	Switzerland	Obtained
Server location	Switzerland	Obtained
Differential privacy	True with epsilon 10	ConfigFile
Method	FedAvg	ConfigFile

Results

The results of the evaluation scenario a of FederatedTrust v.0.2.0 can be seen in Figure 6.4 and Figure 6.2. Figure 6.4 shows the scores of all the pillars with their according notions.

As can be seen, the privacy pillar received a score of 0.49, since differential privacy was applied. However, the chosen *epsilon* of 10 seems to be too small, thus the Indistinguishability notion is scored as zero. The robustness pillar has a score of 0.30 due to the low performance of the model. The fairness pillar received a score of 0.59 since the class distribution is very uneven but the performance of the model among the clients is even. The explainability pillar received a high score of 0.9 and the accountability pillar a score of 0.73. The federation pillar received a score of 0.79, with the notion of client management receiving the highest score of one but the optimization notion a score of 0.57. Coming to the sustainability pillar, the carbon intensity of the energy source notion was scored high with 0.98, since the clients and the server are located in Switzerland which has a low carbon intensity energy grid. The hardware efficiency notion received a score of 0.28. The federation complexity is low and the received score of 0.91 is high, leading to an overall score of the sustainability pillar of 0.79.

The final trust score of this federation is 0.65. Without the newly added sustainability pillar in FederatedTrust v.0.2.0, FederatedTrust v.0.1.0 would have scored the federation with a trust score of 0.63 and is therefore 0.02 points lower than FederatedTrust v.0.2.0 which takes into account the environmental impact a federation has. This showcases that FederatedTrust v.0.2.0 corrects the trust score upwards if the federation has a high environmental impact.

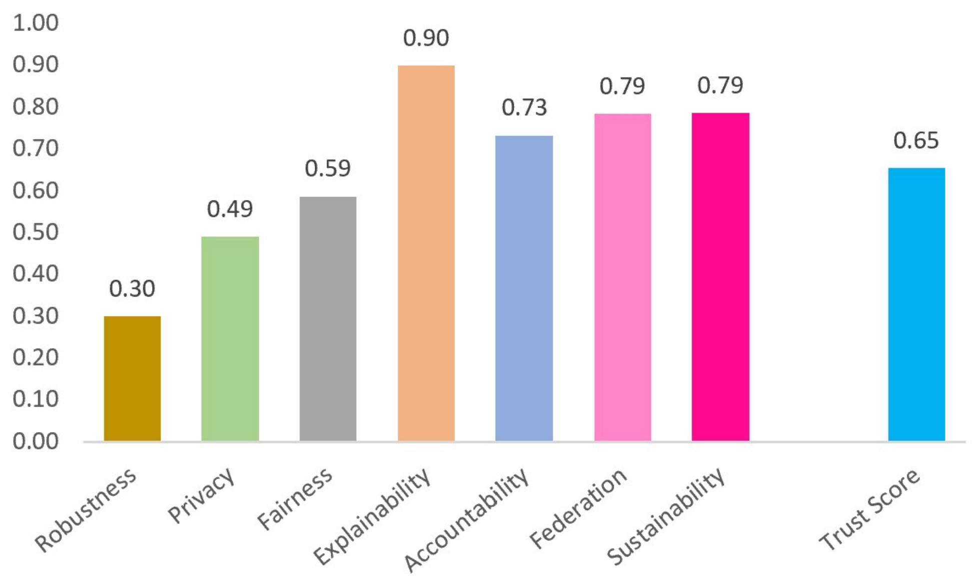


Figure 6.2: Results of Evaluation of FederatedTrust v.0.2.0 Scenario a all Pillars

6.3.2 Scenario b

Scenario Description

The configurations for this evaluation scenario can be viewed in Table 6.8. The federation uses one server and 10e6 clients, all located in South Africa using an Intel Core i7-8650U CPU. The model is a 2-layer CNN and the dataset used is the FEMNIST dataset to classify handwritten digits. The dataset is split into 60% used for training, 20% for testing, and 20% for evaluation. The federation runs for 10e6 global training rounds and clients perform 1000 local training rounds per global training round. In each global training round, 30% of clients are selected to share their model parameters with the server. Differential privacy is on with an *epsilon* of 10. The last column of Table 6.8 shows the source where these metrics have been set. Most of them are set in the ConfigFile of FederatedScope to configure the federation run, some are parametrized in the FactSheet file of FederatedTrust as described in Subsection 5.2.9 to avoid actually running these many training rounds and the hardware that is used by the clients and the server are obtained from the course directly by FederatedTrust.

Table 6.8: Configurations FederatedTrust v.0.2.0 evaluation scenario b

Metric	Value	Source
Model	ConvNet2 (FederatedScope)	ConfigFile
Number of local training rounds	100	FactSheet
Dataset	FEMNIST	ConfigFile
Data split	0.6/0.2/0.2 (train, test, validation)	ConfigFile
Batch size	50	ConfigFile
Loss	CrossEntropyLoss	ConfigFile
Consistent label distribution	False	ConfigFile
Number of clients	10e6	FactSheet
Client Selection rate	30%	ConfigFile
Number of training rounds	10e6	FactSheet
Client hardware	Intel(R) Core(TM) i7-8650U CPU @ 1.90GHz	Obtained
Server hardware	Intel(R) Core(TM) i7-8650U CPU @ 1.90GHz	Obtained
Client location	South Africa	FactSheet
Server location	South Africa	FactSheet
Differential privacy	True with epsilon 10	ConfigFile
Method	FedAvg	ConfigFile

Results

The results of the evaluation scenario b of FederatedTrust v.0.2.0 can be seen in Figure 6.3 and Figure 6.5. Figure 6.5 shows the scores of all the pillars with their according notions.

As can be seen, the privacy pillar received a score of 0.55, and the robustness pillar a score of 0.33 due to the low performance of the model. The fairness pillar received a score of 0.16 since the class distribution is very uneven and also the performance of the model among the clients is uneven. The explainability pillar received a high score of 0.9 and the accountability pillar a score of 0.73. The federation pillar received a score of 0.79, with the notion of client management receiving the highest score of one but the optimization notion a score of 0.57. Coming to the sustainability pillar, the carbon intensity of the energy source notion was scored low with 0.11 as well as the hardware efficiency with a score of 0.28. The federation complexity notion is medium and scored 0.49, since the number of training rounds and clients is high but the model is not complex, leading to an overall score of the sustainability pillar of 0.25.

The final trust score of this federation is 0.53. Without the newly added sustainability pillar in FederatedTrust v.0.2.0, FederatedTrust v.0.1.0 would have scored the federation with a trust score of 0.57 and is therefore 0.04 points higher than FederatedTrust v.0.2.0 which takes into account the environmental impact a federation has. This showcases that FederatedTrust v.0.2.0 corrects the trust score downwards if the federation has a high environmental impact.

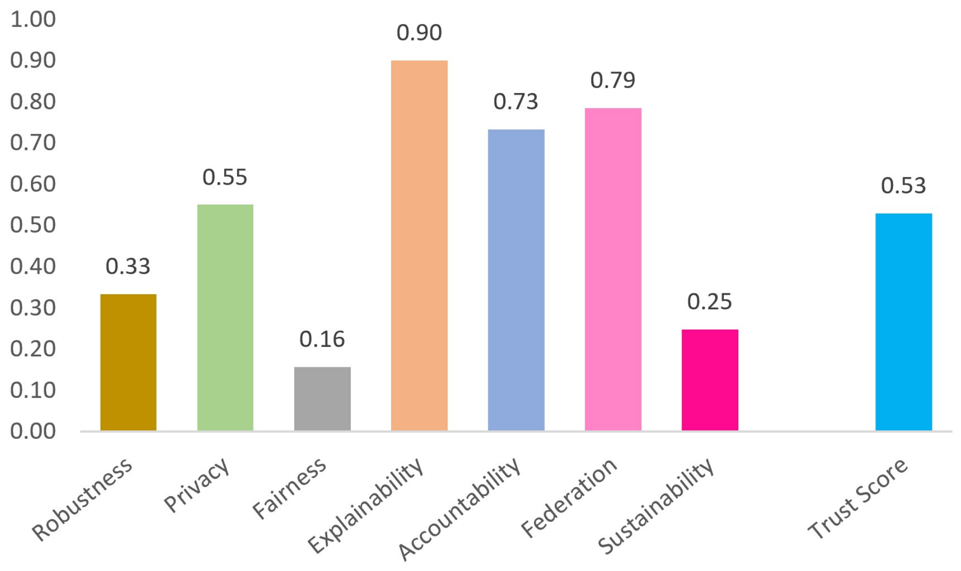


Figure 6.3: Results of Evaluation of FederatedTrust v.0.2.0 Scenario b all Pillars

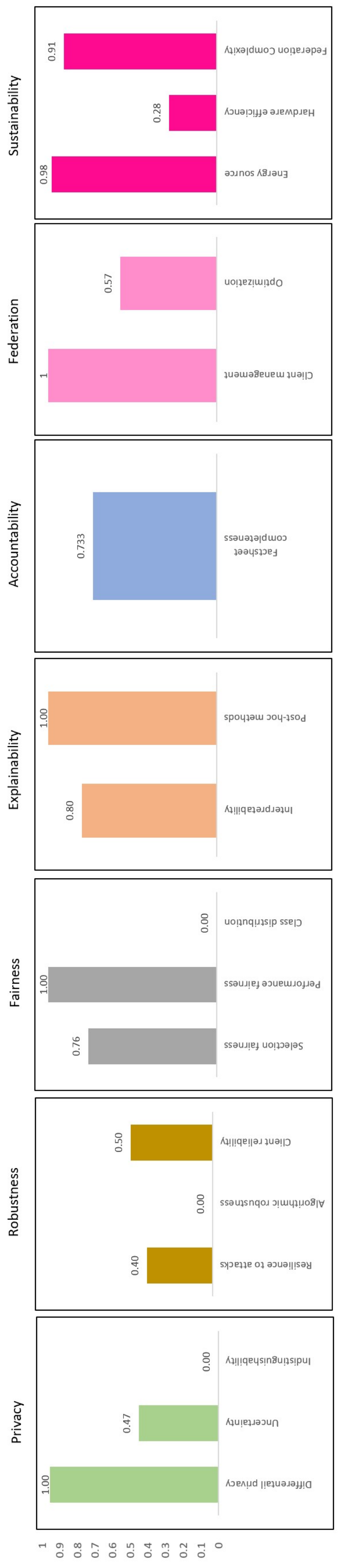


Figure 6.4: Results of Evaluation of FederatedTrust v.0.2.0 Scenario a all Notions

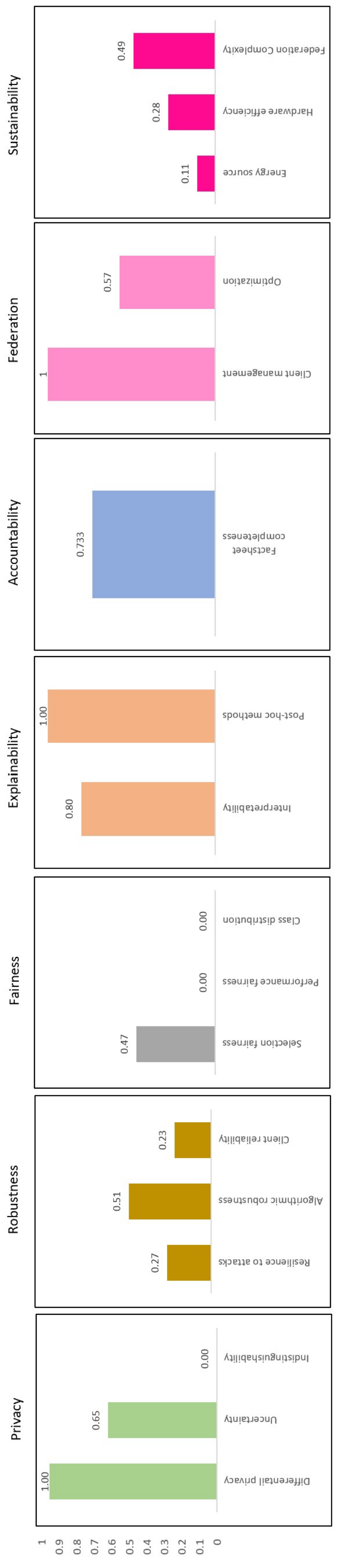


Figure 6.5: Results of Evaluation of FederatedTrust v.0.2.0 Scenario b all Notions

6.4 Discussion and Limitations

The evaluation scenario zero showed that FederatedTrust v.0.2.0 is able to correctly obtain the raw metrics from the added sustainability pillar directly from the federation, its client, and the server using methodologies explained in Subsection 5.2.6. This was shown by running the algorithm once by itself and once passing the raw metrics as parameters in the FactSheet file. Both versions showed the same raw metrics as well as scores for the metrics, notions, and the sustainability pillar as shown in Table 6.1 and Table 6.2. Further, this analysis implies that the four parametrized evaluation scenarios (one, two, three, and four) are valid and their results can be discussed as real evaluation scenarios.

The analysis of the sustainability pillar showed that federations having a small federation complexity and that are using efficient hardware as well as low carbon intensity energy grids receive high scores in the sustainability pillar. This fact was shown in the evaluation scenario one, which received a score of one in the sustainability pillar. This aligns with the expectations of federations with low carbon emissions receiving high scores in the sustainability pillar.

Further, it was shown that federations having a fairly complex federation including large numbers of clients and training rounds as well as big model and dataset sizes, in combination with inefficient hardware and a high carbon intensity energy grid result in low scores in the sustainability pillar. This was shown in evaluation scenario one, which received a score of 0.09 in the sustainability pillar.

Additionally, FL systems having a fairly complex federation as well as inefficient hardware, but using low carbon-intensive energy grids have received neither high nor low scores in the sustainability pillar, but a medium score of 0.65 in the evaluation scenario three. The opposite of this example, being FL systems that have a simple federation and use efficient hardware, but are using a carbon-intensive energy grid also scored medium in the sustainability pillar as shown in evaluation scenario one with a sustainability score of 0.53.

The results of the evaluation scenarios suit the design of the pillar that makes use of the fact that using inefficient hardware and having complex federations lead to higher energy consumption and higher energy consumption leads to higher CO_2eq emissions as well as the fact that the higher the carbon intensity of the energy grid the higher are also the CO_2eq emissions.

The analysis of FederatedTrust v.0.2.0 including the new pillar sustainability and all the six existing pillars shows that FederatedTrust v.0.2.0 provides lower scores for the same federation configuration compared to FederatedTrust v.0.1.0 if the environmental impact of the federation is high and higher scores if the environmental impact is low. This showcases that FederatedTrust v.0.2.0 corrects the trust score of the federation by including the factor of environmental impact. Thus, FederatedTrust v.0.2.0 is the first algorithmic prototype that takes all the seven requirements defined by the AI-HLEG into account and thus closes the gap between research and guidelines defined by governing bodies.

Coming to limitations in terms of the sustainability pillar, the magnitude in which the single metrics influence the CO_2eq emissions are uncertain but are weighted equally. For example, the number of training rounds and the number of clients in the federation have the same weight in this prototype design, but it might be that having more training rounds actually contributes more to the final CO_2eq emissions than the number of clients in the federation or vice-versa. Similarly, on a notion level, it is unclear if the efficiency of the hardware notion and the federation complexity notion influence the CO_2eq emissions equally. Thus, a clear limitation of this prototype design and implementation is the weighting of the metrics and notions, that might not fully reflect the influence they have on the environmental impact of the federation. Further investigations are needed at this point.

Another important aspect to discuss is that since all metrics are weighted the same, if one metric is extraordinarily big, for example, if the number of training rounds is 1 billion, only this one metric is scored low if all the other metrics perform well in the sustainability analysis. This means, that the sustainability pillar would still get a high score, even if a federation with one billion training rounds would probably emit a lot of CO_2eq and is thus not sustainable. Thus, in future works, it might be necessary to think about how to mitigate extremely big metrics.

Similarly in the other six pillars, the influence of the single metrics and notions on the according pillar could be researched and the weights could be adjusted accordingly to achieve a more accurate final trust score.

Currently, the data needed to compute the metrics of the sustainability pillar are stored in plain text, which poses some security risks. The security of the prototype could be enhanced by using encryption for the data that is stored in the emissions file.

As discussed in Section 4.3, the design of the sustainability pillar itself poses some limitations. Privacy-preserving technologies such as DP and HE, often applied in FL, increase the computational costs of a federation and thus also the emissions. Similarly, mechanisms to detect malicious clients might increase the computational costs and the emissions produced. These factors are not yet integrated into the sustainability pillar and could be explored in future works.

Even if the algorithmic prototype has now already implemented 30 metrics, there are still some missing that have been defined in the taxonomy for the pillars robustness, privacy, explainability, fairness, accountability, and federation as discussed in Subsection 3.3.1. In the future, one could enhance the prototype by implementing all metrics and thus provide a more flexible algorithmic prototype that fits a variety of different federation configurations.

Chapter 7

Summary and Conclusion

Artificial intelligence (AI) has immersed our daily lives and assists in the decision process of critical sectors such as medicine and law. Therefore it is now more important than ever before that AI systems developed are reliable, ethical, and do not cause harm to humans. The AI-HLEG [12] has laid the groundwork by defining the seven key requirements for trustworthiness in AI systems: 1) Human Agency and Oversight, 2) Technical Robustness and Safety, 3) Privacy and Data Governance, 4) Transparency, 5) Fairness and non-discrimination, 6) Environmental wellbeing and 7) Accountability. In response to the data privacy risks associated with traditional centralized learning, FL has emerged as a promising alternative. FL allows multiple clients to collaboratively train a global model without the need to share their own private data. Because of the wide adoption of FL, ensuring the trustworthiness of FL systems is crucial.

Early work [14] in this field has proposed a trustworthy FL taxonomy compromising the pillars of privacy, robustness, accountability, explainability, federation, and fairness, each defined with specific notions and metrics. Additionally, prototype algorithms have been implemented to measure the trust level of FL systems. However, to the best knowledge, the requirement of environmental well-being of trustworthy AI/FL has been largely overlooked by research despite being clearly defined by the AI-HLEG. This leaves a clear gap between requirements defined by governing bodies and guidelines applied and measured by research.

This master thesis addresses the gap by introducing the sustainability pillar to the trustworthy FL taxonomy. The sustainability pillar focuses on assessing the environmental impact of FL systems. It consists of three notions: hardware efficiency, federation complexity, and the carbon intensity of the energy grid. The first two notions together define the resource consumption of an FL system, in which a higher federation complexity and less efficient hardware contribute to increased resource consumption. Additionally, the carbon intensity of the energy grid measures CO2eq emissions associated with a kWh of electricity, and thus all the notions together provide insights into the environmental impact of the FL system.

As a second contribution, this master thesis extended the existing prototype for evaluating the trustworthiness of FL systems with an implementation of the sustainability pillar. The implementation uses a Python package called CodeCarbon [90], to obtain the hardware model names used by clients and the server as well as to obtain the carbon intensity of the energy grid used by clients and the server by obtaining their location and taking the carbon intensity of the energy grid used in the country each client or the server is located in.

Extensive evaluations of the prototype across seven different scenarios involving different federation configurations have been conducted. The results of the evaluation scenarios show, that FL systems of low complexity, efficient hardware, and a low carbon-intensive energy grid as the one of Albania, receive a high score in the sustainability pillar of trustworthy FL. On the contrary, complex federations using inefficient hardware and a carbon-intensive energy grid receive low scores in the sustainability pillar. Further, complex federations that use inefficient hardware but a low-carbon-intensive energy grid as well as simple federations using efficient hardware but a carbon-intensive energy grid receive medium scores around 0.5. Further, the evaluation scenarios including all the seven pillars show that this extended version of FederatedTrust provides lower trust scores for federations having high environmental impact and higher trust scores for federations having low environmental impact compared to FederatedTrust v.0.1.0 that did not take the environmental impact into account. This showcases that FederatedTrust v.0.2.0 corrects the trust score of the federation by including the factor of environmental impact and thus provides a more comprehensive trust score by including seven pillars representing the seven requirements defined by the AI-HLEG.

In conclusion, this master's thesis contributes to the research community by providing the first trustworthy FL taxonomy that includes all the seven requirements defined by the AI-HLEG. Moreover, it introduces the first algorithmic prototype, that evaluates the trustworthiness level of FL systems including all the seven requirements. Ultimately, the integration of sustainability into trustworthy FL taxonomies closes the gap between requirements defined by governing bodies and research.

7.1 Future Work

In future work, the contribution of individual metrics to carbon emissions can be explored and used to fine-tune the weighting of the metrics to achieve more precise sustainability scores. Similarly, the contribution of metrics to the other pillars could be explored and fine-tuned with weights. Further, other aspects of a federation that influence the emissions could be included such as the computational costs for privacy-preserving methodologies such as DP and HE as well as the computational costs for malicious client detection such as clustering or the H-MINE algorithm. The security of the prototype could be enhanced by encrypting information stored at the side of FederatedTrust, Despite all the metrics defined in the sustainability pillar being implemented in the prototype, the algorithmic prototype could be enhanced by incorporating missing metrics of the other six pillars that are defined in the taxonomy but not yet implemented in the prototype. As DFL is emerging, the prototype could be adjusted to work with decentralized federations in the future. Currently, the prototype only works combined with the FederatedScope framework, thus future work could integrate it into other frameworks. These changes could enhance the prototype and improve its flexibility to different federations using different methodologies, different frameworks as well as decentralized architectures.

Declaration of Independence for Written Work

Selbständigkeitserklärung

Hiermit erklr̈e ich, dass ich die vorliegende Arbeit selbstn̈dig und ohne Benutzung anderer als der angegebenen Hilfsmittel (inklusive generativer KI wie z.B. ChatGPT) angefertigt habe. Mir ist bekannt, dass ich die volle Verantwortung für die Wissenschaftlichkeit des vorgelegten Textes selbst übernehme, auch wenn (nach schriftlicher Absprache mit der betreuenden Professorin resp. dem betreuenden Professor) KI-Hilfsmittel eingesetzt und deklariert wurden. Alle Stellen, die wörtlich oder sinngems̈s aus veröffentlichten oder nicht veröffentlichten Schriften entnommen wurden, sind als solche kenntlich gemacht. Die Arbeit ist in gleicher oder ḧnlicher Form oder auszugsweise im Rahmen einer anderen Prüfung noch nicht vorgelegt worden.

[illegible]

Naters, den August 7, 2023 ....

Statement of authorship

I hereby declare that I have composed this work independently and without the use of any aids other than those declared (including generative AI such as ChatGPT). I am aware that I take full responsibility for the scientific character of the submitted text myself, even if AI aids were used and declared (after written confirmation by the supervising professor). All passages taken verbatim or in sense from published or unpublished writings are identified as such. The work has not yet been submitted in the same or similar form or in excerpts as part of another examination.

[illegible]

Naters, August 7, 2023 ....

Bibliography

[1] S. D. Holcomb, W. K. Porter, S. V. Ault, G. Mao, and J. Wang, “Overview on deepmind and its alphago zero ai,” in *Proceedings of the 2018 international conference on big data and education*, 2018, pp. 67–71.

[2] M. B. Hoy, “Alexa, siri, cortana, and more: an introduction to voice assistants,” *Medical reference services quarterly*, vol. 37, no. 1, pp. 81–88, 2018.

[3] C. A. Gomez-Uribe and N. Hunt, “The netflix recommender system: Algorithms, business value, and innovation,” *ACM Transactions on Management Information Systems (TMIS)*, vol. 6, no. 4, pp. 1–19, 2015.

[4] OpenAI, “Chatgpt,” https://openai.com/blog/chatgpt, accessed: 03.02.2023.

[5] P. G. Mikhael, J. Wohlwend, A. Yala, L. Karstens, J. Xiang, A. K. Takigami, P. P. Bourgouin, P. Chan, S. Mrah, W. Amayri *et al.*, “Sybil: A validated deep learning model to predict future lung cancer risk from a single low-dose chest computed tomography,” *Journal of Clinical Oncology*, vol. 41, no. 12, pp. 2191–2200, 2023.

[6] I. Ajunwa, S. Friedler, C. E. Scheidegger, and S. Venkatasubramanian, “Hiring by algorithm: predicting and preventing disparate impact,” *Available at SSRN*, 2016.

[7] C. McKay, “Predicting risk in criminal procedure: actuarial tools, algorithms, ai and judicial decision-making,” *Current Issues in Criminal Justice*, vol. 32, no. 1, pp. 22–39, 2020.

[8] Z. Obermeyer and S. Mullainathan, “Dissecting racial bias in an algorithm that guides health decisions for 70 million people,” in *Proceedings of the conference on fairness, accountability, and transparency*, 2019, pp. 89–89.

[9] A. Brackey, “Analysis of racial bias in northpointe’s compas algorithm,” Ph.D. dissertation, Tulane University School of Science and Engineering, 2019.

[10] J. Dastin, “Amazon scraps secret ai recruiting tool that showed bias against women,” in *Ethics of data and analytics.* Auerbach Publications, 2018, pp. 296–299.

[11] N. H. T. S. Administration, “Motor vehicle traffic crash data resource,” https://crashstats.nhtsa.dot.gov/#!/#%2F, accessed: 10.03.2023.

[12] A. HLEG, “thics guidelines for trustworthy ai,” https://ec.europa.eu/futurium/en/ai-alliance-consultation.1.html, accessed: 15.02.2023.

[13] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[14] P. M. S. Sánchez, A. H. Celdrán, N. Xie, G. Bovet, G. M. Pérez, and B. Stiller, "Federatedtrust: A solution for trustworthy federated learning," *arXiv preprint arXiv:2302.09844*, 2023.

[15] A. R. Elkordy, J. Zhang, Y. H. Ezzeldin, K. Psounis, and S. Avestimehr, "How much privacy does federated learning with secure aggregation guarantee?" *arXiv preprint arXiv:2208.02304*, 2022.

[16] 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.

[17] E. T. M. Beltrán, M. Q. Pérez, P. M. S. Sánchez, S. L. Bernal, G. Bovet, M. G. Pérez, G. M. Pérez, and A. H. Celdrán, "Decentralized federated learning: Fundamentals, state-of-the-art, frameworks, trends, and challenges," 2022. [Online]. Available: https://arxiv.org/abs/2211.08413

[18] T. D. Nguyen, T. Nguyen, P. L. Nguyen, H. H. Pham, K. Doan, and K.-S. Wong, "Backdoor attacks and defenses in federated learning: Survey, challenges and future research directions," *arXiv preprint arXiv:2303.02213*, 2023.

[19] Y. Liu, Y. Kang, T. Zou, Y. Pu, Y. He, X. Ye, Y. Ouyang, Y.-Q. Zhang, and Q. Yang, "Vertical federated learning," *arXiv preprint arXiv:2211.12814*, 2022.

[20] S. AbdulRahman, H. Tout, H. Ould-Slimane, A. Mourad, C. Talhi, and M. Guizani, "A survey on federated learning: The journey from centralized to distributed on-site learning and beyond," *IEEE Internet of Things Journal*, vol. 8, no. 7, pp. 5476–5497, 2020.

[21] Alexsoft, "Federated learning: The shift from centralized to distributed on-device model training," https://www.altexsoft.com/blog/federated-learning, accessed: 13.02.2023.

[22] L. He, A. Bian, and M. Jaggi, "Cola: Decentralized linear learning," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[23] E. T. M. Beltrán, M. Q. Pérez, P. M. S. Sánchez, S. L. Bernal, G. Bovet, M. G. Pérez, G. M. Pérez, and A. H. Celdrán, "Decentralized federated learning: Fundamentals, state-of-the-art, frameworks, trends, and challenges," *arXiv preprint arXiv:2211.08413*, 2022.

[24] B. Li, P. Qi, B. Liu, S. Di, J. Liu, J. Pei, J. Yi, and B. Zhou, "Trustworthy ai: From principles to practices. arxiv 2021," *arXiv preprint arXiv:2110.01167*, 2022.

[25] H. Liu, Y. Wang, W. Fan, X. Liu, Y. Li, S. Jain, Y. Liu, A. Jain, and J. Tang, "Trustworthy ai: A computational perspective," *ACM Transactions on Intelligent Systems and Technology*, vol. 14, no. 1, pp. 1–59, 2022.

[26] A. Brintrup, G. Baryannis, A. Tiwari, S. Ratchev, G. Martinez-Arellano, and J. Singh, "Trustworthy, responsible, ethical ai in manufacturing and supply chains: synthesis and emerging research questions," *arXiv preprint arXiv:2305.11581*, 2023.

[27] I. Goodfellow, Y. Bengio, and A. Courville, "Machine learning basics (chapter 5)(2016)," *Deep Learning (MIT Press, 95–151)*, 2016.

[28] L. Rocher, J. M. Hendrickx, and Y.-A. De Montjoye, "Estimating the success of re-identifications in incomplete datasets using generative models," *Nature communications*, vol. 10, no. 1, pp. 1–9, 2019.

[29] Z. Zhou, F. Sun, X. Chen, D. Zhang, T. Han, and P. Lan, "A decentralized federated learning based on node selection and knowledge distillation," *Mathematics*, vol. 11, no. 14, p. 3162, 2023.

[30] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan, "A survey on bias and fairness in machine learning," *ACM Computing Surveys (CSUR)*, vol. 54, no. 6, pp. 1–35, 2021.

[31] S. Barocas, A. Guo, E. Kamar, J. Krones, M. R. Morris, J. W. Vaughan, W. D. Wadsworth, and H. Wallach, "Designing disaggregated evaluations of ai systems: Choices, considerations, and tradeoffs," in *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, 2021, pp. 368–378.

[32] S. Caton and C. Haas, "Fairness in machine learning: A survey," *arXiv preprint arXiv:2010.04053*, 2020.

[33] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, "A survey of methods for explaining black box models," *ACM computing surveys (CSUR)*, vol. 51, no. 5, pp. 1–42, 2018.

[34] A. Rosenfeld and A. Richardson, "Explainability in human–agent systems," *Autonomous Agents and Multi-Agent Systems*, vol. 33, pp. 673–705, 2019.

[35] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins *et al.*, "Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai," *Information fusion*, vol. 58, pp. 82–115, 2020.

[36] D. Leslie, "Understanding artificial intelligence ethics and safety," *arXiv preprint arXiv:1906.05684*, 2019.

[37] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *2019 IEEE symposium on security and privacy (SP)*. IEEE, 2019, pp. 691–706.

[38] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," *Advances in neural information processing systems*, vol. 32, 2019.

[39] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," in *2019 IEEE symposium on security and privacy (SP)*. IEEE, 2019, pp. 739–753.

[40] L. Lyu, H. Yu, X. Ma, C. Chen, L. Sun, J. Zhao, Q. Yang, and S. Y. Philip, "Privacy and robustness in federated learning: Attacks and defenses," *IEEE transactions on neural networks and learning systems*, 2022.

[41] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE symposium on security and privacy (SP)*. IEEE, 2017, pp. 3–18.

[42] B. Zhao, K. R. Mopuri, and H. Bilen, "idlg: Improved deep leakage from gradients," *arXiv preprint arXiv:2001.02610*, 2020.

[43] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the gan: information leakage from collaborative deep learning," in *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, 2017, pp. 603–618.

[44] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, 2015, pp. 1322–1333.

[45] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology—EUROCRYPT'99: International Conference on the Theory and Application of Cryptographic Techniques Prague, Czech Republic, May 2–6, 1999 Proceedings 18*. Springer, 1999, pp. 223–238.

[46] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the forty-first annual ACM symposium on Theory of computing*, 2009, pp. 169–178.

[47] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE transactions on information theory*, vol. 31, no. 4, pp. 469–472, 1985.

[48] W. Jin, Y. Yao, S. Han, C. Joe-Wong, S. Ravi, S. Avestimehr, and C. He, "Fedml-he: An efficient homomorphic-encryption-based privacy-preserving federated learning system," *arXiv preprint arXiv:2303.10837*, 2023.

[49] W. Zhang, Y. Zhao, F. Li, and H. Zhu, "A hierarchical federated learning algorithm based on time aggregation in edge computing environment," *Applied Sciences*, vol. 13, no. 9, p. 5821, 2023.

[50] D. Enthoven and Z. Al-Ars, "An overview of federated deep learning privacy attacks and defensive strategies," *Federated Learning Systems: Towards Next-Generation AI*, pp. 173–196, 2021.

[51] C. Xie, K. Huang, P.-Y. Chen, and B. Li, "Dba: Distributed backdoor attacks against federated learning," in *International conference on learning representations*, 2020.

[52] H. Wang, K. Sreenivasan, S. Rajput, H. Vishwakarma, S. Agarwal, J.-y. Sohn, K. Lee, and D. Papailiopoulos, "Attack of the tails yes, you really can backdoor federated learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 16 070–16 084, 2020.

[53] 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.

[54] C. Fung, C. J. Yoon, and I. Beschastnikh, "The limitations of federated learning in sybil settings." in *RAID*, 2020, pp. 301–316.

[55] 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.

[56] Z. Liu, J. Guo, W. Yang, J. Fan, K.-Y. Lam, and J. Zhao, "Privacy-preserving aggregation in federated learning: A survey," *IEEE Transactions on Big Data*, 2022.

[57] M. Moshawrab, M. Adda, A. Bouzouane, H. Ibrahim, and A. Raad, "Reviewing federated learning aggregation algorithms; strategies, contributions, limitations and future perspectives," *Electronics*, vol. 12, no. 10, p. 2287, 2023.

[58] A. Z. Tan, H. Yu, L. Cui, and Q. Yang, "Towards personalized federated learning," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[59] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[60] Y. Mansour, M. Mohri, J. Ro, and A. T. Suresh, "Three approaches for personalization with applications to federated learning," *arXiv preprint arXiv:2002.10619*, 2020.

[61] Y. Zhan, J. Zhang, Z. Hong, L. Wu, P. Li, and S. Guo, "A survey of incentive mechanism design for federated learning," *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 2, pp. 1035–1044, 2021.

[62] Y. Wang and B. Kantarci, "A novel reputation-aware client selection scheme for federated learning within mobile environments," in *2020 IEEE 25th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD).* IEEE, 2020, pp. 1–6.

[63] S. Abdul Rahman, "Adaptive client selection and upgrade of resources for robust federated learning," Ph.D. dissertation, École de technologie supérieure, 2022.

[64] 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.

[65] Y. Jiang, W. Zhang, and Y. Chen, "Data quality detection mechanism against label flipping attacks in federated learning," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 1625–1637, 2023.

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

[67] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *ICC 2019-2019 IEEE international conference on communications (ICC)*. IEEE, 2019, pp. 1–7.

[68] M. Ribero and H. Vikalo, "Communication-efficient federated learning via optimal client sampling," *arXiv preprint arXiv:2007.15197*, 2020.

[69] J. Goetz, K. Malik, D. Bui, S. Moon, H. Liu, and A. Kumar, "Active federated learning," *arXiv preprint arXiv:1909.12641*, 2019.

[70] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing federated learning on non-iid data with reinforcement learning," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 1698–1707.

[71] N. Yoshida, T. Nishio, M. Morikura, K. Yamamoto, and R. Yonetani, "Hybrid-fl for wireless networks: Cooperative learning mechanism using non-iid data," in *ICC 2020-2020 IEEE International Conference On Communications (ICC)*. IEEE, 2020, pp. 1–7.

[72] Q. Zeng, Y. Du, K. Huang, and K. K. Leung, "Energy-efficient resource management for federated edge learning with cpu-gpu heterogeneous computing," *IEEE Transactions on Wireless Communications*, vol. 20, no. 12, pp. 7947–7962, 2021.

[73] P. Zhou, P. Fang, and P. Hui, "Loss tolerant federated learning," *arXiv preprint arXiv:2105.03591*, 2021.

[74] M. Mohri, G. Sivek, and A. T. Suresh, "Agnostic federated learning," in *International Conference on Machine Learning*. PMLR, 2019, pp. 4615–4625.

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

[76] L. Lyu, J. Yu, K. Nandakumar, Y. Li, X. Ma, J. Jin, H. Yu, and K. S. Ng, "Towards fair and privacy-preserving federated deep models," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 11, pp. 2524–2541, 2020.

[77] A. Abay, Y. Zhou, N. Baracaldo, S. Rajamoni, E. Chuba, and H. Ludwig, "Mitigating bias in federated learning," *arXiv preprint arXiv:2012.02447*, 2020.

[78] 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.

[79] N. Xie, "Quantifying the trustworthiness level of federated learning models," *Available at https://www.merlin.uzh.ch/publication/show/22995*, November 2022.

[80] S. Rabiul Islam, W. Eberle, and S. K. Ghafoor, "Towards quantification of explainability in explainable artificial intelligence methods," *arXiv e-prints*, pp. arXiv–1911, 2019.

[81] S. K. Lo, Q. Lu, L. Zhu, H.-Y. Paik, X. Xu, and C. Wang, "Architectural patterns for the design of federated learning systems," *Journal of Systems and Software*, vol. 191, p. 111357, 2022.

[82] L. Fu, H. Zhang, G. Gao, H. Wang, M. Zhang, and X. Liu, "Client selection in federated learning: Principles, challenges, and opportunities," *arXiv preprint arXiv:2211.01549*, 2022.

[83] A. Tariq, M. A. Serhani, F. Sallabi, T. Qayyum, E. S. Barka, and K. A. Shuaib, "Trustworthy federated learning: A survey," *arXiv preprint arXiv:2305.11537*, 2023.

[84] A. S. Luccioni and A. Hernandez-Garcia, "Counting carbon: A survey of factors influencing the emissions of machine learning," *arXiv preprint arXiv:2302.08476*, 2023.

[85] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in nlp," *arXiv preprint arXiv:1906.02243*, 2019.

[86] A. S. Luccioni, S. Viguier, and A.-L. Ligozat, "Estimating the carbon footprint of bloom, a 176b parameter language model," *arXiv preprint arXiv:2211.02001*, 2022.

[87] D. Patterson, J. Gonzalez, Q. Le, C. Liang, L.-M. Munguia, D. Rothchild, D. So, M. Texier, and J. Dean, "Carbon emissions and large neural network training," *arXiv preprint arXiv:2104.10350*, 2021.

[88] A. S. George, A. H. George, and A. G. Martin, "The environmental impact of ai: A case study of water consumption by chat gpt," *Partners Universal International Innovation Journal*, vol. 1, no. 2, pp. 97–104, 2023.

[89] X. Qiu, T. Parcollet, J. Fernandez-Marques, P. P. de Gusmao, Y. Gao, D. J. Beutel, T. Topal, A. Mathur, and N. D. Lane, "A first look into the carbon footprint of federated learning." *J. Mach. Learn. Res.*, vol. 24, pp. 129–1, 2023.

[90] CodeCarbon, "Codecarbon," https://codecarbon.io, accessed: 22.02.2023.

[91] P. Henderson, J. Hu, J. Romoff, E. Brunskill, D. Jurafsky, and J. Pineau, "Towards the systematic reporting of the energy and carbon footprints of machine learning," 2020.

[92] A. Lacoste, A. Luccioni, V. Schmidt, and T. Dandres, "Quantifying the carbon emissions of machine learning," *arXiv preprint arXiv:1910.09700*, 2019.

[93] D. Patterson, J. Gonzalez, U. Hölzle, Q. Le, C. Liang, L.-M. Munguia, D. Rothchild, D. R. So, M. Texier, and J. Dean, "The carbon footprint of machine learning training will plateau, then shrink," *Computer*, vol. 55, no. 7, pp. 18–28, 2022.

[94] S. Schlömer, T. Bruckner, L. Fulton, E. Hertwich, A. McKinnon, D. Perczyk, J. Roy, R. Schaeffer, R. Sims, P. Smith *et al.*, "Annex iii: Technology-specific cost and performance parameters," in *Climate change 2014: Mitigation of climate change: Contribution of working group III to the fifth assessment report of the Intergovernmental Panel on Climate Change.* Cambridge University Press, 2014, pp. 1329–1356.

[95] B. Petroleum, "Statistical review of world energy 2022. bp," 2022.

[96] O. W. in Data, "Carbon intensity of electricity," https://ourworldindata.org/grapher/carbon-intensity-electricity, accessed: 03.07.2023.

[97] M. Martonosi, D. Brooks, and P. Bose, "Modeling and analyzing cpu power and performance: Metrics, methods, and abstractions," *SIGMETRICS 2001/Performance 2001-Tutorials*, 2001.

[98] PassMark, "Benchmarking," https://www.cpubenchmark.net/, accessed: 25.03.2023.

[99] ——, "Thermal design power," https://www.cpubenchmark.net/power_performance.html, accessed: 03.05.2023.

[100] ——, "Cpu benchmark dataset," https://www.kaggle.com/datasets/alanjo/cpu-benchmarks, accessed: 03.05.2023.

[101] ——, "Gpu benchmark dataset," https://www.kaggle.com/datasets/alanjo/gpu-benchmarks, accessed: 03.05.2023.

[102] Y. Xie, Z. Wang, D. Gao, D. Chen, 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.

[103] G. Kamiya, "The carbon footprint of streaming video: fact-checking the headlines," 2020.

Abbreviations

AI	Artificial Intelligence
AI-HLEG	High-Level Expert Group on Artificial Intelligence
BP	British Petrolium
CFL	Centralized Federated Learning
CNN	Convolutional Neural Network
CO_2eq	Carbon Dioxide equivalents
COMPAS	Correctional Offender Management Profiling for Alternative Sanctions
CPU	entral Processing Unit
DFL	Decentralized Federated Learning
DL	Deep Learning
DNN	DNN Deep Neural Network
DP	Differential Privacy
FAFL	Fairness Aware Federated Learning
FedAvg	FederatedAveraging
FL	Federated Learning
FLOPS	Floating-Point Operations Per Second
GAN	Generative Adversarial Network
GPT-3	Generative Pre-trained Transformer 3
GPT-4	Generative Pre-trained Transformer 4
GPU	Graphics Processing Unit
HE	Homomorphic Encryption
HFL	Horizontal Federated Learning
IPCC	Intergovernmental Panel on Climate Change
IPS	Instructions Per Second
ML	Machine Learning
MNN	Multi-layer Neural Network
NLP	Natural Language Processing
US	United States
RNN	Recurrent Neural Network
SMC	Secure Multiparty Computation
SVM	Support Vector Machine
TDP	Thermal Design Power
VFL	Vertical Federated Learning
XAI	Explainable AI

List of Figures

List of Tables

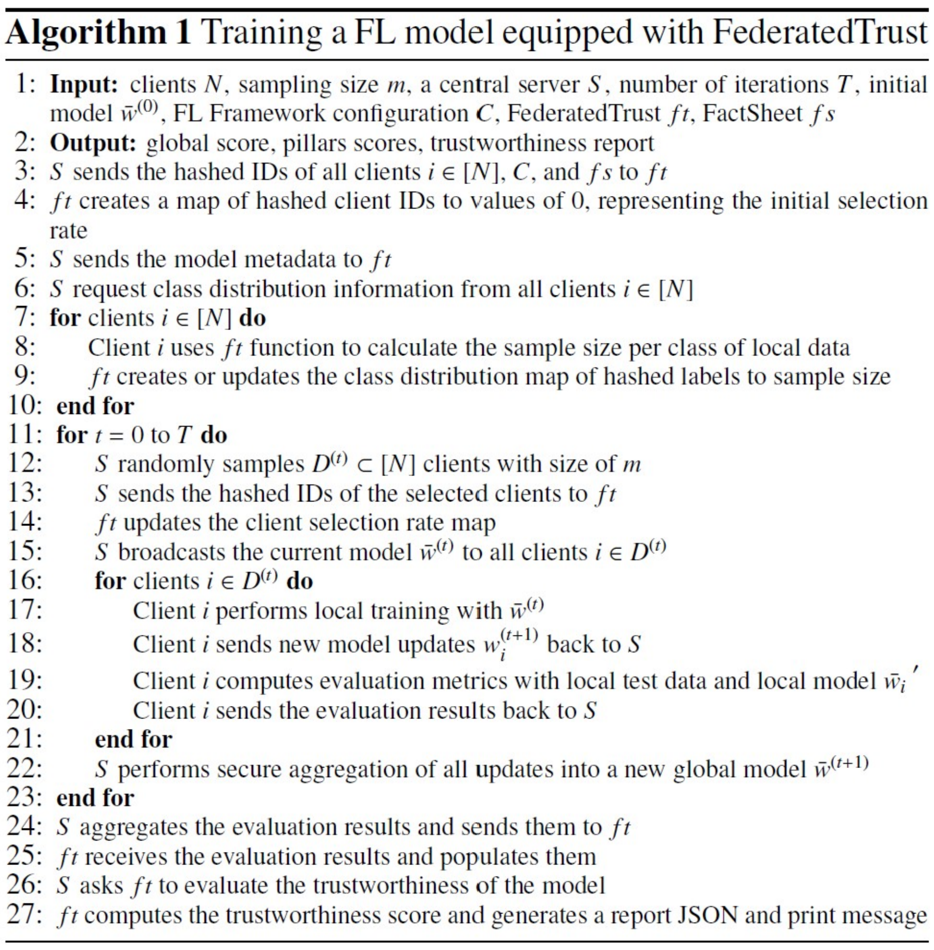
Code Listings

List of Algorithms

Appendix A

FederatedTrust v.0.1.0

A.1 Algorithmic Pseudocode

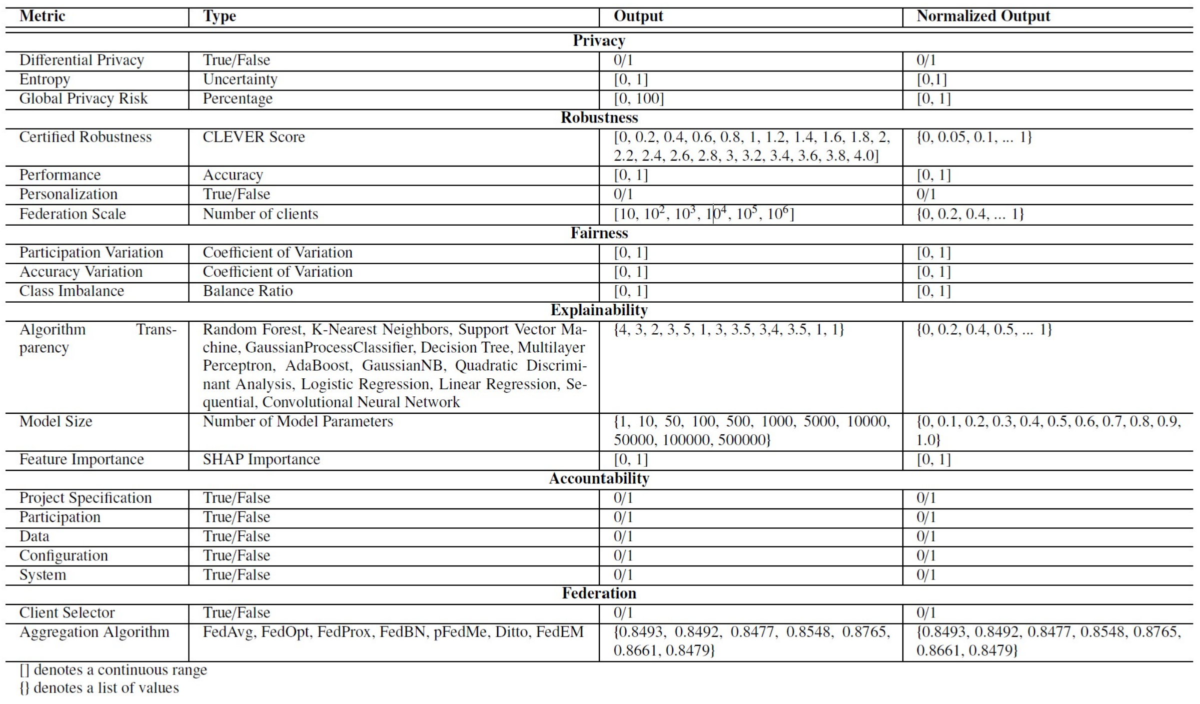


Algorithm 1 Training a FL model equipped with FederatedTrust

```
1: Input: clients $N$, sampling size $m$, a central server $S$, number of iterations $T$, initial
   model $\bar{w}^{(0)}$, FL Framework configuration $C$, FederatedTrust $ft$, FactSheet $fs$
2: Output: global score, pillars scores, trustworthiness report
3: $S$ sends the hashed IDs of all clients $i \in [N]$, $C$, and $fs$ to $ft$
4: $ft$ creates a map of hashed client IDs to values of 0, representing the initial selection
   rate
5: $S$ sends the model metadata to $ft$
6: $S$ request class distribution information from all clients $i \in [N]$
7: for clients $i \in [N]$ do
8:     Client $i$ uses $ft$ function to calculate the sample size per class of local data
9:     $ft$ creates or updates the class distribution map of hashed labels to sample size
10: end for
11: for $t = 0$ to $T$ do
12:     $S$ randomly samples $D^{(t)} \subset [N]$ clients with size of $m$
13:     $S$ sends the hashed IDs of the selected clients to $ft$
14:     $ft$ updates the client selection rate map
15:     $S$ broadcasts the current model $\bar{w}^{(t)}$ to all clients $i \in D^{(t)}$
16:     for clients $i \in D^{(t)}$ do
17:         Client $i$ performs local training with $\bar{w}^{(t)}$
18:         Client $i$ sends new model updates $w_i^{(t+1)}$ back to $S$
19:         Client $i$ computes evaluation metrics with local test data and local model $\bar{w}_i{}'$
20:         Client $i$ sends the evaluation results back to $S$
21:     end for
22:     $S$ performs secure aggregation of all updates into a new global model $\bar{w}^{(t+1)}$
23: end for
24: $S$ aggregates the evaluation results and sends them to $ft$
25: $ft$ receives the evaluation results and populates them
26: $S$ asks $ft$ to evaluate the trustworthiness of the model
27: $ft$ computes the trustworthiness score and generates a report JSON and print message
```

Figure A.1: FederatedTrust Algorithm Pseudocode from [14].

A.2 Normalization Functions for Metrics



Metric	Type	Output	Normalized Output
Privacy			
Differential Privacy	True/False	0/1	0/1
Entropy	Uncertainty	[0, 1]	[0,1]
Global Privacy Risk	Percentage	[0, 100]	[0, 1]
Robustness			
Certified Robustness	CLEVER Score	[0, 0.2, 0.4, 0.6, 0.8, 1, 1.2, 1.4, 1.6, 1.8, 2, 2.2, 2.4, 2.6, 2.8, 3, 3.2, 3.4, 3.6, 3.8, 4.0]	{0, 0.05, 0.1, ... 1}
Performance	Accuracy	[0, 1]	[0, 1]
Personalization	True/False	0/1	0/1
Federation Scale	Number of clients	[10, 10^2, 10^3, 10^4, 10^5, 10^6]	{0, 0.2, 0.4, ... 1}
Fairness			
Participation Variation	Coefficient of Variation	[0, 1]	[0, 1]
Accuracy Variation	Coefficient of Variation	[0, 1]	[0, 1]
Class Imbalance	Balance Ratio	[0, 1]	[0, 1]
Explainability			
Algorithm Transparency	Random Forest, K-Nearest Neighbors, Support Vector Machine, GaussianProcessClassifier, Decision Tree, Multilayer Perceptron, AdaBoost, GaussianNB, Quadratic Discriminant Analysis, Logistic Regression, Linear Regression, Sequential, Convolutional Neural Network	{4, 3, 2, 3, 5, 1, 3, 3.5, 3,4, 3.5, 1, 1}	{0, 0.2, 0.4, 0.5, ... 1}
Model Size	Number of Model Parameters	{1, 10, 50, 100, 500, 1000, 5000, 10000, 50000, 100000, 500000}	{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0}
Feature Importance	SHAP Importance	[0, 1]	[0, 1]
Accountability			
Project Specification	True/False	0/1	0/1
Participation	True/False	0/1	0/1
Data	True/False	0/1	0/1
Configuration	True/False	0/1	0/1
System	True/False	0/1	0/1
Federation			
Client Selector	True/False	0/1	0/1
Aggregation Algorithm	FedAvg, FedOpt, FedProx, FedBN, pFedMe, Ditto, FedEM	{0.8493, 0.8492, 0.8477, 0.8548, 0.8765, 0.8661, 0.8479}	{0.8493, 0.8492, 0.8477, 0.8548, 0.8765, 0.8661, 0.8479}

[] denotes a continuous range
{} denotes a list of values

Figure A.2: FederatedTrust Normalization of Metrics from [14].