Université de Montréal

Towards Causal Federated Learning : A Federated Approach To Learning Representations Using Causal Invariance

par

Sreya Francis

Département de mathématiques et de statistique
Faculté des arts et des sciences

Mémoire présenté en vue de l'obtention du grade de
Maître ès sciences (M.Sc.)
en Discipline

October 26, 2021

Université de Montréal

Faculté des arts et des sciences

Ce mémoire intitulé

Towards Causal Federated Learning : A Federated Approach To Learning Representations Using Causal Invariance

présenté par

Sreya Francis

a été évalué par un jury composé des personnes suivantes :

Guy Wolf

(président-rapporteur)

Irina Rish

(directeur de recherche)

Golnoosh Farnadi

(membre du jury)

Résumé

L'apprentissage fédéré est une approche émergente d'apprentissage automatique distribué préservant la confidentialité pour créer un modèle partagé en effectuant une formation distribuée localement sur les appareils participants (clients) et en agrégeant les modèles locaux en un modèle global. Comme cette approche empêche la collecte et l'agrégation de données, elle contribue à réduire dans une large mesure les risques associés à la vie privée. Cependant, les échantillons de données de tous les clients participants sont généralement pas indépendante et distribuée de manière identique (non-i.i.d.), et la généralisation hors distribution (OOD) pour les modèles appris peut être médiocre. Outre ce défi, l'apprentissage fédéré reste également vulnérable à diverses attaques contre la sécurité dans lesquelles quelques entités participantes malveillantes s'efforcent d'insérer des portes dérobées, dégradant le modèle agrégé généré ainsi que d'inférer les données détenues par les entités participantes. Dans cet article, nous proposons une approche pour l'apprentissage des caractéristiques invariantes (causales) communes à tous les clients participants dans une configuration d'apprentissage fédérée et analysons empiriquement comment elle améliore la précision hors distribution (OOD) ainsi que la confidentialité du modèle appris final. Bien que l'apprentissage fédéré permette aux participants de contribuer leurs données locales sans les révéler, il se heurte à des problèmes de sécurité des données et de paiement précis des participants pour des contributions de données de qualité. Dans ce rapport, nous proposons également une conception et un flux de travail EOS Blockchain pour établir la sécurité des données, une nouvelle métrique basée sur les erreurs de validation sur laquelle nous qualifions les téléchargements de gradient pour le paiement, et implémentons un petit exemple de notre modèle d'apprentissage fédéré blockchain pour analyser ses performances.

Mots clés: apprentissage fédéré, apprentissage automatique causal, blockchain, généralisation hors distribution, apprentissage automatique préservant la confidentialité, robustesse, equité

Abstract

Federated Learning is an emerging privacy-preserving distributed machine learning approach to building a shared model by performing distributed training locally on participating devices (clients) and aggregating the local models into a global one. As this approach prevents data collection and aggregation, it helps in reducing associated privacy risks to a great extent. However, the data samples across all participating clients are usually not independent and identically distributed (non-i.i.d.), and Out of Distribution (OOD) generalization for the learned models can be poor. Besides this challenge, federated learning also remains vulnerable to various attacks on security wherein a few malicious participating entities work towards inserting backdoors, degrading the generated aggregated model as well as inferring the data owned by participating entities. In this work, we propose an approach for learning invariant (causal) features common to all participating clients in a federated learning setup and analyse empirically how it enhances the Out of Distribution (OOD) accuracy as well as the privacy of the final learned model. Although Federated Learning allows for participants to contribute their local data without revealing it, it faces issues in data security and in accurately paying participants for quality data contributions. In this report, we also propose an EOS Blockchain design and workflow to establish data security, a novel validation error based metric upon which we qualify gradient uploads for payment, and implement a small example of our Blockchain Causal Federated Learning model to analyze its performance with respect to robustness, privacy and fairness in incentivization.

Keywords: Federated Learning, Causal Machine Learning, Blockchain, Out-of-Distribution Generalization, Privacy Preserving Machine Learning, Robustness, Fairness

Table des matières

Liste des tableaux

Liste des figures

Liste des sigles et des abréviations

FL	*Federated Learning*
DML	*Distributed Machine Learning*
IRM	*Invariant Risk Minimization*
IPFS	*Inter Planetary File System*
ERM	*Empirical Risk Minimization*
RMatch	*Random Match*

Remerciements

I would first like to thank my supervisor, Professor Irina Rish, whose expertise was invaluable in formulating the research questions and methodology. Her insightful feedback pushed me to sharpen my thinking. Her kindness, compassion and empathy taught me what it means to be a good leader and changed me for the better.

I would like to acknowledge my colleagues from medical group at MILA for their helpful collaboration and for all of the opportunities I was given to further my research. I also thank all my teachers for their valuable guidance throughout my studies and for providing me with the tools that I needed to choose the right direction at each phase of my research.

Most importantly, I would like to thank my grand-father (Thomas), my parents (Susan and Francis) and my best-friend Nivedita for being my constant pillars of support. I also acknowledge the support of my extended family and friends who provided stimulating discussions as well as happy distractions to rest my mind outside of my research.

Chapitre 1

Introduction

Federated learning is an approach to Distributed Machine Learning developed by the Google AI team; this approach allows users to keep ownership of their data during the model training process. In addition to keeping ownership of their data, users also have immediate access to the newest model after they have trained it on their data. The sender of the model benefits from distributed data, lower latency, and less local power consumption. Since Federated Learning ensures privacy, more users who don't share their data currently become more willing to partake in the training process and ultimately creating smarter models.

The training process involves a central server sending out a model to a subset of the current users. This set downloads the model, trains the model on their device with their local data, and returns only the update that resulted in the new model. Each device returns their particular update and the central server aggregates these updates to improve the original model, which is the new global model.

1.1. Thesis Statement

In a federated setting, the data samples across all participating clients are usually not independent and identically distributed (non-i.i.d.), and Out of Distribution (OOD) generalization for the learned models can be poor. Besides this challenge, federated learning also remains vulnerable to various attacks on security wherein a few malicious participating entities work towards inserting backdoors, degrading the generated aggregated model as well as inferring the data owned by participating entities. In this work, we propose an approach for learning invariant (causal) features common to all participating clients in a federated learning setup and analyse empirically how it enhances the Out of Distribution (OOD) accuracy as well as the privacy of the final learned model. Moreover we also propose an approach to record and reward contributions of each participant with the help of blockchain on our system implementation.

1.2. Statement Of Contributions

The main contribution of this thesis are as follows:

- This thesis presents some approaches to bridge the fields of invariant causal feature learning and federated learning.
- Despite the focus of existing work on enhancing test performance for clients within the federated setting, the area of study on how these FL algorithms can generalize well to unseen testing clients is relatively under explored. Our study helps assess how FL performance can be enhanced in an out of distribution test setting for unseen test clients.
- We show that our federated causal invariance learning approach helps enhance not just the out of distribution generalization performance but also ensures higher security against some of the privacy attacks posed in a distributed setting.
- This thesis also includes an attempt to bridge the fields of blockchain and federated learning for fair incentivization.
- Our work with blockchain proposes an approach to democratize access to data and enhance fair incentivization to every participant in the federated training process.
- This thesis also includes one of the very first attempts to bridge the fields of causal invariance learning, federated learning and blockchain.

1.3. List Of Papers Included In The Thesis:

- Towards Causal Federated Learning For Enhanced Robustness and Privacy (37) Presented at ICLR 2021 Distributed and Private Machine Learning Workshop.
- Record and Reward Federated Learning Contributions With Blockchain (78) Presented at IEEE Conference on Distributed Computing and Knowledge Discovery.
- CausalFedBlock: Blockchain For Fair Incentivization in Causal Federated Learning Accepted to AAAI 2022 Trustworthy AI for Healthcare Workshop.

Certain aspects of this thesis are taken from works that are in preparation for publication or have been published. In particular, the introduction includes modified parts of (37) and (78). The majority of Chapter 3 is reproduced from (37). The majority of Chapter 4 is reproduced from (78) and chapter 5 is in preparation for publication. The author of this thesis was the lead author of all these works, and the collaborators acknowledge the use of these works in this thesis.

Chapitre 2

Related Work

The project at hand seeks to study causal approaches to enhance Federated Learning.

Our objective of this paper is to review the state-of-the-art of systems similar to our proposed topic of interest. These state-of-the-art solutions will be reviewed across three separate sections – Section 2.1: Federated Learning, Section 2.2: Distributed Machine Learning (DML), 5.2.4: Blockchain and Section 2.7: Causal Learning. To the best of our knowledge, this report is the first to investigate causal federated learning.

2.1. Federated Learning

Federated Learning is an approach to DML developed by the Google AI team (83)(57); this approach allows users to keep ownership of their data during the model training process. In addition to keeping ownership of their data, users also have immediate access to the newest model after they have trained it on their data. The sender of the model benefits from distributed data, lower latency, and less local power consumption. Since Federated Learning ensures privacy, more users who don't share their data currently become more willing to partake in the training process and ultimately creating smarter models.

The training process involves a central server sending out a model to a subset of the current users. This set downloads the model, trains the model on their device with their local data, and returns only the update that resulted in the new model. Each device returns their particular update and the central server aggregates these updates to improve the original model,which is the new global model (83)(57)(56).

This is an approach (57) (83) whereby one entity $\mathcal{O}$ owns the training model T but not the data; the full dataset is distributed among many users in U. This model is trained in the following way:

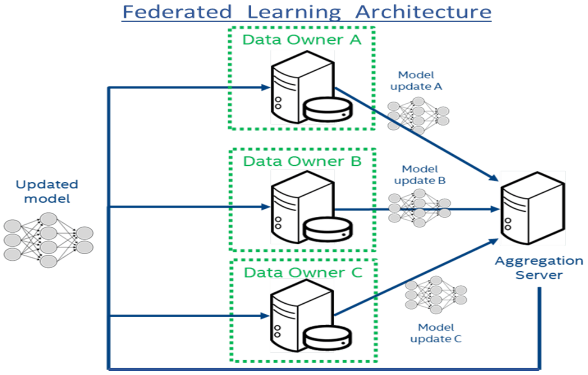


Fig. 2.1. Federated Learning Architecture based on client server model (114)

(1) $\mathcal{O}$ distributes the same model T to many users $u \in U$.
(2) Each user u feeds their local data through the model T, and calculates their gradient update δ (57) to improve the model performance.
(3) Each user u returns their update value δ to $\mathcal{O}$.
(4) $\mathcal{O}$ receives the collection of update values δ and averages these values to a single value $\bar{\delta}$ (57).
(5) $\mathcal{O}$ applies $\bar{\delta}$ to the current model T to obtain the new global model T', and $T \leftarrow T'$.
(6) Steps 1-5 are repeated until $\mathcal{O}$ stops the Federated Learning process, at which point training is complete.

The process can be broken down in the following way – we have multiple machines where each maintains a copy of the training model, but keeps their local dataset as model input representing a subset of the global dataset. These machines share the same parameters of the training model, by uploading and downloading parameters to and from a centralized parameter server. Each machine then upload their local training gradients based on which the training model is updated by using some optimizer like Stochastic Gradient Descent (SGD). The machines download updated parameters from the parameter server and continue to train the local model. This process repeats until machines obtain the final trained model (120).

2.1.1. Privacy Threats

Although the Federated Learning architecture and process makes considerable efforts to keep the user's data private, an attacker could analyze the weights of the sent updates to make conclusions about the data of users (103). Certain Machine Learning algorithms such as Neural Networks and Recurrent Language models are known to memorize data labelling and patterns. In such cases, a user's data may risk losing its privacy since they are represented in the model (15). While this might sound unlikely if not done on purpose, there have been experiments that show it is possible to reconstruct some data points (38)

FL algorithms are vulnerable to some attacks, namely membership inference (7) (102), model inversion (80) and model extraction (36).
Membership Inference typically determine whether a point is in the training dataset or not. Shokri et al. (102) propose a shadow training technique: first train k shadow models to mimic the behavior of target model, then accordingly train an attack (membership inference) model. Salem et al. (7) greatly broaden the application of (102) by gradually relaxing its limitations: first adversary only needs to train one shadow model; second adversary uses a totally different training set and one shadow model; then third adversary could infer the membership only based on empirical statistics.
Model Inversion attacks try to use black-box access to estimate the feature values from training dataset. Fredrikson et al. (80) explored model inversion attacks in two settings: decision trees and neural networks.
Model Extraction attacks try to duplicate the parameters of target model. Tramèr et al. (36) propose effective attack methods to logistic regression, neural networks and decision trees.

Model Poisoning, which differs from traditional data poisoning is one of the major threats to a federated setting which involves an adversary controlling a small number of malicious agents, mostly not exceeding one in number, aiming to cause the global model to misclassify a set of chosen inputs with very high confidence hence degrading the global model performance.

Membership Inference Attack which is yet another threat to federated setting was first presented by Shokri et al (102) . The general idea behind this attack is to use several models (one for each prediction class), called as attack models, to make membership inference over the target model's output which is its posterior probabilities.Assuming that the target model is a black-box API, the proposition was to construct several such shadow models to mimic the target model's behavior and derive the required data which is again the posterior.

Adversarial Robustness We observed that despite many relevant work on adversarial defense, most of them fail when it comes to generalizable robustness to adversarial examples beyond distributional constraints(95). The most approached method is to include adversarial examples in the training cohort. We observe that causal feature learning(95) can help beyond existing pre-defined norms.

2.1.2. Defense Strategies

To combat this possible privacy issue, McMahan, Daniel & Kunal have proposed a differentially private variant of federated learning framework (103) (15). This variant proposes changes to the Federation Learning algorithm in order to provide differential privacy to a client's contributions. This is achieved by randomly sampling user updates to be aggregated, and adding Gaussian noise to the final updates of this group (15)(38)(103). In spite of high computation cost and slower convergence rate, this tends to provide a negligible loss of accuracy as the previous state-of-the-art implementations. (15) (38)

Some researchers have proposed some defense strategies for robust federated learning ((26), (97)) FoolsGold (26) is a defense mechanism against Sybil attacks by adjusting the learning rates of local models based on contribution similarity. The algorithm identifies grouped actions as Sybil attacks and promotes the difference of local model update. However, FoolsGold may identify harmless participants as attackers when these participants have similar local data. (138) proposed a model, CalTrain, that represents data with fingerprints to identify poisoned data and models. (58) propose to maintain a small reference dataset to justify the quality and accountability of models. While this method is effective, it requires a lot of time to evaluate each model in every single round. Some researchers proposed to improve the privacy preservation of federated learning (103) (121) There has also been several proposals for a privacy-preserving protocol for model aggregation in federated learning. (103) was the first to introduce differential privacy into federated learning.

By using cryptography techniques, it is possible to ensure that the updates of individuals can only be read when enough users has submitted updates (13). This makes man-in-the-middle attacks much harder, i.e. an adversary cannot make conclusions about the training data based on the intercepted network activity of an individual user. To perform such an attack, an adversary would need to intercept the messages of many users (13).

2.1.3. Domain Shift Issues

The main idea behind federated learning is to have each node learn on its own local data and not share either the data or the model parameters.While federated learning promises better privacy and efficiency, existing methods ignore the fact that the data on each node are collected in a non-i.i.d manner, leading to domain shift between nodes (52). For example, one device may take photos mostly indoors,while another mostly outdoors.
Consider two clients A and B in a Federated learning setup. In real word scenario, we always have $P_A \neq P_B$ The participating clients can have varying marginal distribution $\mathcal{P}_A(x)$ termed covariate shift despite of having a shared $\mathcal{P}(B \mid x)$. Apart from feature distribution skew, there can also be cases with label distribution skew wherein marginal distributions $\mathcal{P}_A(y)$ may vary across clients, even if $\mathcal{P}(x \mid B)$ is the same (95)

2.1.4. Computational Assumptions

Federated Learning was first proposed with two possible avenues for decreased smartphone computation – either use more devices to decrease computation amount per phone, or have each phone partake in more complex computations between model updates. Both of these proposals are based on the assumption that phone processing power can easily handle the computations necessary for model updates during an idle state (45).
In the event that the amount of computation needed is too large for the phone or that a phone is never idle enough to perform these computations, the use of local computation devices – such as in Mobile Edge Computing –is needed (72).

2.2. Distributed Machine Learning

Generally, there are two approaches for distributed machine learning – model parallelism and data parallelism. The former partitions a training model among multiple machines with the same dataset, and the latter partitions the dataset among multiple machines with the same model (83)(120). In the case of Federated Learning, we focus on the data parallelism approach.

One problem that may arise is the *disclosure* of local data and model. Though each machine uploads only its gradient which should keep its local data hidden, adversaries can infer important information about this local data by initiating inference attacks or membership attacks (71).

2.2.1. Federated Learning Vs Distributed Machine Learning

While Federated Learning and DML are similar on a technical level, Federated Learning has some major differences to DML applications in data centers where the training data is distributed among many machines (57)(111). Since Machine Learning generally requires a lot of data, many applications for which Machine Learning models are generated have many users; every one of these users could theoretically participate in Federated Learning, making it far more distributed than anything in a data center. While earlier DML setting expected data to be balanced with identical distribution, Federated Learning should expect unbalanced number of samples with non identical distributions. (57). In a DML setting, it is expected that nodes can communicate quickly with each other and that it is ensured that messages do not get lost; in Federated Learning, these assumptions cannot be made. Uploads are typically much slower than downloads and may be extremely slow – especially in the case of mobile phone networks (124)(57)(111).

Federated learning is not just a distributed version of standard machine learning. It is a distributed system and therefore must be robust to arbitrarily misbehaving participants. Unfortunately, existing techniques for Byzantine-tolerant distributed learning do not apply when the participants' training data are not i.i.d., which is exactly the motivating scenario for federated learning. How to design robust federated learning systems is an important topic for future research.

2.3. Causal Machine Learning

In contrast to the current domain generalization approaches used in federated learning, causal invariant learning techniques to address OOD generalization problem in unseen test client distribution are derived from the causal inference literature which targets use of causal variables for prediction problems.

All causal learning studies assume access to heterogeneous data from multiple environments which is exactly the scenarion in a federated learning setting. In our approach, we concentrate on 'causal invariance' as underlying principle under the assumption that if predictors are causal, estimator is invariant across interventions.

2.3.1. Data Heterogenity

In a federated setting consider that the data of each client is denoted by $D^c = \{X^c, Y^c\}$ which represents multiple training environments $c \in \operatorname{supp}(\mathcal{E}_{train})$, $X^c \subset \mathcal{X}$ and $Y^c \subset \mathcal{Y}$ are the collection of data and label from each client environment c. Let P^c be the distribution of data and label in client environment c. Usually, for all $c \in \operatorname{supp}(\mathcal{E}_{train})$, the data and label distribution $P^c(X, Y)$ can be quite different from that of training environments $\mathcal{E}_{train}$

2.3.2. Common Assumptions Made In Causal Invariance Learning

2.3.2.1 Assumption On Causality:

In causal inference literature, it common to make an assumption about causally invariant relationship between the target Y and its direct causes $X_{\mathrm{pa}(Y)}$ which can be an indicator on the stability of causal variables $X_{\mathrm{pa}(Y)}$ across different participating client environments (17).

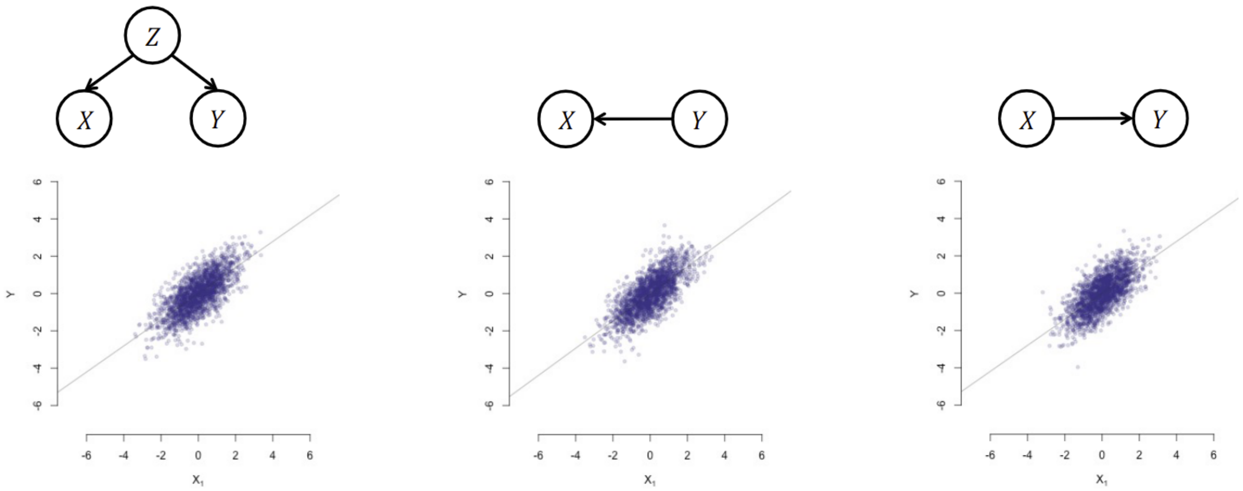


Fig. 2.2. Structural Equation Models can generate same observational distribution in many cases

More specifically, this assumption of causality states that given the structural equation models:

$$Y^c \leftarrow f_Y\left(X_{pa(Y)}^c, \epsilon_Y^c\right), \epsilon_Y^c \perp X_{pa(Y)}^c,$$

remains the same across all environments $c \in \operatorname{supp}\left(\mathcal{E}_{\text{all}}\right)$, that is, ϵ_Y^c has the same distribution as ϵ_Y for all client environments(115).

Several methods have been developed aiming to obtain causal variables from heterogeneous data to achieve OOD generalization. However as the main standard to such cause effect identification is via randomized control trials, these approaches are not practical in real world scenarios or usual machine learning settings. Hence the research focus was shifted to obtaining a causal explanation to find prensence of invariance across different data environments. Several papers (98) (28) (47) (40)(92) have studied methods to leverage such invariance in participating data environmnets.

2.3.2.2 Assumption On Invariance

The invariance assumption states that there exists a subset $S^* \subseteq \{1, \ldots, p\}$ of the covariate indices such that $P\left(Y^c \mid X_{S^*}^c\right)$ is the same for all participating client environments $c \in \mathcal{E}$. This implies that the conditional distribution is invariant across all participating client environments from $\mathcal{E}$ when conditioning on the covariates from S^*(115).

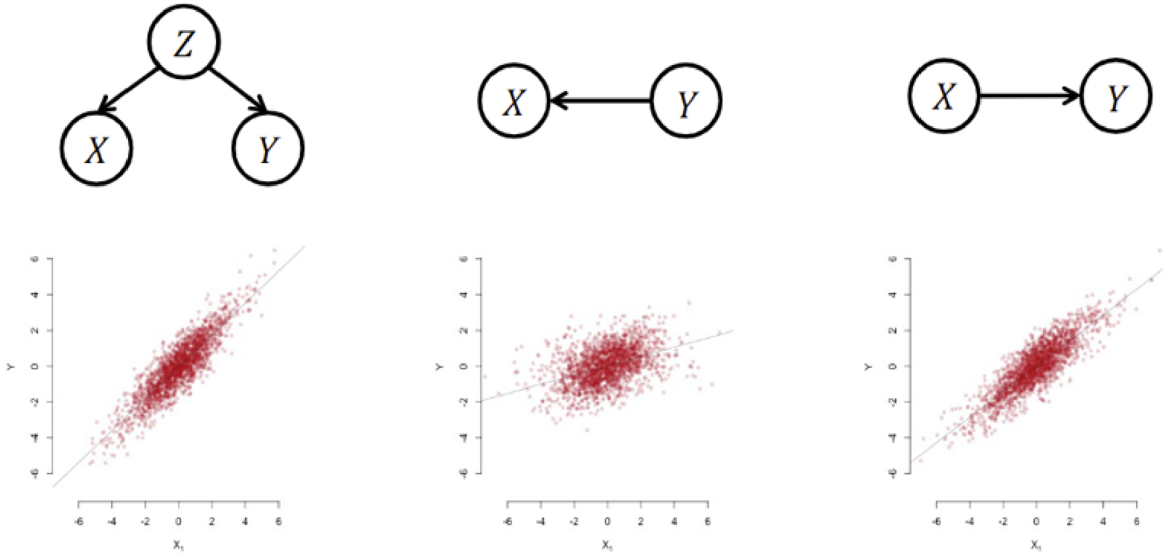


Fig. 2.3. Structural Equation Models can generate same observational distribution in some cases under shift interventions on X

The initial study on invariance being a possible factor that can help infer causal structure under certain conditions and assumtions was done by (96) in Invariant Causal Predcition with a detailed statistical test on whether a subset of covariates can satisfy the invariance assumption. In this paper (96), they state that the conditional distribution of the target given the direct causes will remain constant when interfering all variables except for the target in the model when all its direct causes are considered.

2.4. Invariant Risk Minimization

(96) was extended to more practical and general settings by invariant risk minimization (10) method which mainly targets latent causal mechanisms and generalizes the invariance assumption to representation level.

(10) assumes that there exists data representation $\Phi(X)$ such that for all client envrionment $, c^{\prime} \in \operatorname{supp}\left(\mathcal{E}_{train}\right), \mathbb{E}\left[Y \mid \Phi\left(X^{c}\right)\right]=\mathbb{E}\left[Y \mid \Phi\left(X^{c^{\prime}}\right)\right]$, where $\mathcal{E}_{\text{train}}$ can be considered the available client train environments.

Based on this assumption, (10) aims to find data representation $\Phi(X)$ that can evoke an invariant linear predictor w across all client train environments $\mathcal{E}_{train}$(115).

$$\begin{aligned} &\min_{\Phi(X), w} \sum_{c \in \operatorname{supp}\left(\mathcal{E}_{\text{train}}\right)} \mathcal{L}^{c}(w \odot \Phi(X), Y) \\ &\text{s.t. } w \in \arg\min_{\bar{w}} \mathcal{L}^{c}(\bar{w} \odot \Phi(X)), \text{ for all } c \in \operatorname{supp}\left(\mathcal{E}_{\text{train}}\right) \end{aligned}$$

2.5. What consist of a Fully Fledged FL Ecosystem?

The proposed inclusion of causal learning techniques in federated learning setting explores a novel FL model training schemes, which is especially effective in non-IID learning scenarios. Apart from the model training, the establishment of a complete Federated Learning system requires a series of auxiliary mechanisms which includes incentive mechanism (108)(54), model compression(22), backdoor defense (58), communication protocol (14)(99), etc., which will guarantee the healthy development of FL ecosystem. We have found that our proposed approaches can be seamlessly embedded into most of the existing methods on these topics but we specifically concentrate on incentivization schemes to begin with in this thesis. Hence, we will discuss some main topics in blockchain that can help with incentivization schemes in our proposed FL setting.

2.6. Blockchain For Incentivization

Blockchain is a revolutionary technology that has the potential to cast a great impact on modern society with its transparency, decentralization, and security properties. This technology gained considerable attention due to its very first application of Cryptocurrencies like Bitcoin. In the near future, Blockchain technology is determined to transform the way we live, interact, and perform business transactions.

Blockchain can be considered as the chain of digital blocks connected and associated with each other as an open distributed ledger which was initially used to store only transactions of digital currencies but later started being widely used in numerous applications beyond currency and payments (118).

There are 3 main categories of Blockchains solely based on their usage and distinct attributes as follows:

- Public/Permissionless blockchains : Blockchains are truly decentralized and allow anyone to join the network and engage in managing them.
- Private/Permissioned blockchains : Blockchains that allow only invited people from a single organization to join the network and manage them.
- Consortium blockchains : These Federated Blockchains have attributes in between public and private Blockchain, in terms of permissions and management. It allows only invited people from different organizations to join the network.

Now we will take a look at some of the basic concepts associated with blockchain technology:

Peer to Peer/ P2P Network : P2P network is a distributed network architecture which enables sharing of resources among participants who make their resources available to be shared with other participants. Each participant node is considered a 'peer' and acts in roles of both client and server. At one time, peer C, acting as client can directly request services from other peer B who acts as server of the network without any intermediate entities and vice versa. (113).

Hash And Hash Chain :

Hash is one-way mathematical function in which original data cannot be calculated back from the unique output to protect the integrity of data. It works by calculating a fixed-sized unique value called hash value for every variable input.(112)

A hash chain is generated by successively applying the hash function on a piece of data and plays an integral role in Blockchain(61)

Cryptography And Encryption : There are two types of modern cryptography, namely Symmetric key cryptography in which same key is used by sender and receiver for cryptographic operations and Asymmetric key cryptography where each communicating party has two different keys called public and private keys used for different cryptographic operations in different ways. (8) Encryption is a process to encode the plaintext into cipher text. The decryption is the reverse process to convert cipher text into plaintext.(112)

Digital Signatures And TimeStamp : Digital signatures are used as a proof of authorship along with the contents. The signatures are usually applied using public key cryptography in which, a signer uses its private key to sign a document and a recipient can verify the signatures using signer's public key.(85) Timestamp is the time at which event occurrence is recorded by a computer and it records the date and time of the day at which event occurred which is accurate to a small fraction of second. This timestamp's data is recoded in a consistent manner along with the actual data for easy comparison of two different records to track progress over time.(12) (34)

2.7. How Can Blockchain Help?

Blockchain is a distributed, decentralized, immutable ledger used to store encrypted data whereas Federated learning can enable analytics and decision making from the large groups of data collected from diversified clients.

It goes without saying that each technology has its own individual degree of complexity, but both Federated Learning and Blockchain are in situations where they can benefit from each other, and help one another.

With both these technologies able to effect and enact upon data in different ways, their coming together makes sense, and it can take the exploitation of data to new levels. At the same time, the integration of Federated Learning into Blockchain, and vice versa, can enhance blockchain's underlying architecture and boost potential of Federated Learning.

Additionally, Blockchain can also make Federated learned models more coherent and understandable, and we can trace and determine why decisions are made in each of them. Blockchain and its ledger can record all data and variables that go through a decision made under Federated Learning.

Putting the two technologies together has the potential to use data in ways never before thought possible. Data is the key ingredient for the development and enhancement of all Federated learning algorithms, and Blockchain secures this data, allows us to audit all intermediary steps taken to draw conclusions from the data and allows participating clients to monetize their produced data.

In our attempt, we mainly concentrate on rewarding participating client contributions as well as enhancing privacy.
The main features of blockchain that can help in our attempt of developing an incentivization system are listed below:

- **Decentralization** The Blockchain ledger exists on multiple computers/devices, called nodes, that form a Blockchain network by working in a Peer to peer mechanism, validating access to the information without a centralized authority (130)(132)(70) . A distributed structure is followed for storing, updating, recording, transmission, verification and maintenance in the Blockchain network (67)(131). This specific feature of blockchain eliminates the need for powerful central authorities paving the way to transfer of control to the individual user which will help in making the system fair and considerably more secure. Most of the transactions are validated using a set of rules and algorithms called consensus protocols(which is achieved when majority devices agree about what should be recorded onto a Blockchain) to ensure that information is consistent and incorruptible (130) (137).
- **Transparency** All transactions lmade on Blockchain are entirely with open availability of details and history of any transaction to anyone. This unique feature of Blockchain provides a great deal of accountability and integrity to the information, ensuring zero deceitful alteration/addition/ deletion.
- **Security** The use of asymmetrical cryptography in blockchain which includes a set of public keys visible to anyone and a set of private keys visible only to the owner, makes this technology incredibly secure. The public and private keys ensure the authenticity, integrity, confidentiality, and authorization of the transaction (35) (131) (130).
- **Immutability** This unique charactersitic of blockchain guarantees that any source of data that has been added to a Blockchain (35), (130) will remain un-altered, un-tampered (131), persistent (137) and unforgeable (70). Each data block within

a Blockchain is time stamped as well as encrypted with hash algorithm ensuring immutability of each data entry. (67), (131). As none of these transactions can be changed or deleted, this process is irreversible and immutable with any change leading to generation of a different hash right away (100) (130).

- **Anonymity** Anonymity is one of the main characteristics of Blockchain that helps ensure privacy. This particular feature is achieved by authenticating transactions without revealing any personal information of parties involved in the transaction ensuring protection from unauthorized intrusion or observation (9) (67) (137).
- **Democratization** Blockchain follows a P2P approach to ensure democracy in all decisions made by pariticipating nodes, that are relatively independent with equal rights, with the help of consensus algorithms (130).Consensus gives permission to specific nodes to add or append new blocks and ensures proper synchronization of their copies across all nodes in the blockchain (132), (130), (131).

2.8. EOS Blockchain

EOSIO is the open-source blockchain led by the company Block.one. The cryptocurreny EOS runs on this blockchain. EOSIO is comparable to the larger currency and blockchain Ethereum in that its blockchain enables other smart contracts and decentralized apps in addition to its own currency. But unlike Ethereum, EOS transactions don't require any fees(44) (5). EOSIO has been designed to address scalability and latency issues in the pre-existing blockchains like Ethereum. It is built like an operating system on which applications are built. The software provides accounts, authentication, databases, asynchronous communication, and the scheduling of applications across many of CPU cores or clusters.
A very detailed description of EOS can be found in the EOS white paper(44).

2.8.1. DPOS - Delegated Proof Of Stake

EOS doesn't rely on a proof-of-work (PoW) mining system which is the main system used in Bitcoin and Ethereum. Instead, EOS is based on delegated proof-of-stake (DPOS) system which relies on block producers voted on by the network to handle the blockchain operations on its behalf(5). Proof-of-Work mining system involves the miners solving a numerical puzzle which is computationally intensive and in return, miners receive a fixed number of bitcoins as reward. In such a system the entities with the highest computation resources are expected to succeed in adding a new block. Additionally such a system encourages users to form mining pools and thereby causing consolidation of computation power within large pools which begins to create an environment of centralisation.

In a significant improvement over the proof-of-work system, the proof-of-stake system eliminates the compute-intensive competition and instead selects validators of transactions in a block from the nodes in a network using an algorithm based on the stake (an amount of cryptocurrency proposed by the node) and coin-age of the nodes. The coin-age conveys how long the node has stayed as a validator. Nodes with a higher coin-age and higher stake value will be preferred. The stake along with the block reward is handed to the validator once the new block with the verified transactions are approved by the other nodes in the network. The current validator's coin-age is set to 0. Even in the POS system, centralization can happen over time if a group of candidate nodes decide to combine their wealth, enabling them to raise higher stakes thereby improving their chances of being picked by the algorithm.

The Delegated POS improves the method of selection of validators by making it more democratic and reducing the advantage of wealth. Users vote for a delegate to become block producer by sending tokens to a pool assigned to the delegate. A group of block producers are elected and they are scheduled to produce blocks in an order agreed upon by the producers. Once a block is produced, the block reward from verifying the transactions are shared among the voters who elected for the particular block producer based on each voter's stake or token value. This system produces blocks in a series of rounds with one block being produced every 0.5 seconds under normal conditions. A transaction can be considered confirmed with 99.9% certainty after an average of 0.25 seconds from time of broadcast(44).

2.8.2. Benefits

While EOS centralizes some of the blockchain operations more than competitors, it also enables better scale and higher transaction volumes than some others. Bitcoin network fees and Ethereum gas can be expensive, and transaction times can slow down during periods of network congestion. EOS aims to provide a faster transaction time with no built-in fees. As EOS relies on fewer computers and hence consume less energy to keep the network running, it is considered more environment friendly(5).
The main advantages of EOS blockchain are specified below:

- It can support thousands of transactions per second which is drastically higher than what Bitcoin or Ethereum can support.
- Parallel processing is supported.
- The usage of DPOS makes it much more energy efficient than the other platforms.
- EOSIO has been designed for scalability and hence this can be achieved readily by adding more computing power.
- There are no built-in fees.

- There are provisions for handling unpredictable or undesired behaviour, wherein the corresponding smart contract or application can be frozen or modified based on a vote by the block producers

Chapitre 3

Federated Causal Invariance Learning

This chapter partly consists of a reproduction of (37). This was accepted to ICLR 2021 Distributed and Private Machine Learning Workshop with reviewers rating this a very promising and plausible research direction.
The idea was initially conceptualized by the author of this thesis as a work on enabling learning representations using causal invariance in a distributed learning setting. The author of this thesis is the primary author and is responsible for coming up with the idea, formulating the problem statement, developing the architecture, experimentation and analysis. Her advisor Prof. Irina Rish provided advising and helped refine the paper.

Federated Learning enables participating entities to learn a shared prediction model in collaboration while keeping their training data locally. We know that Federated Learning prevents data collection and aggregation and hence helps in reducting associated privacy risks to a great extent. However, it still remains vulnerable to numerous attacks on security wherein a few malicious participating entities work towards:

- degrading the generated aggregated model
- inserting backdoors
- inferring the data owned by participating entities

In this chapter we explore the ways in which Causal feature learning can enhance the out of distribution robustness and various ways in which it can help reduce effects of backdoor as well as inference attacks in federated learning setup. Also federated approach to causal learning has added advantages to the field of causal learning. So this is a two way beneficial approach.

3.1. Motivation

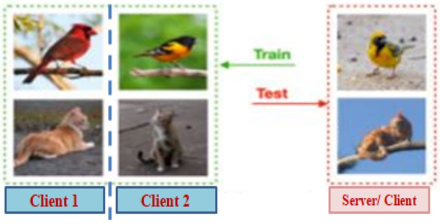


Fig. 3.1. Client data distribution in usual federated learning setting

It is next to impossible to observe client level data to be distributed in an independent and identical manner in a federated learning setup. When training a federated learning system with the i.i.d. assumption, an implicit assumption is made on the underlying data generating process defining an envrionment for the client level data. Each data generating processes leads to different client environments consisting of varying underlying distributions of features and targets.

If the server side environment or client side test environment in which inference is to be done differs from the environment in which the federated learning model was trained, we get poor results owing to the fact that correlations between features and the target are different. To be more specific, the federated model can only map from features to target in one environment resulting in inferring incorrect values of the target for the features in a different server/client environment.

Current associational learning techniques used in federated learning aims to exploit correlations between features to gain predictive performance and encodes the invariants of the dataset on which they're trained. Birds and branches are highly correlated due to which if birds dominantly appear on the dataset, we should expect this to be learned.

Our goal is to develop a federated model highly confident of predictions outside the client level train datasets with enhanced robustness to distributional shifts away from the training set with weightage to learning a representation common to all participating clients relying upon features that affect the target in all participating client environments ignoring individual client dataset-specific correlations. We propose approaches to develop a federated model that generalizes beyond the limited set of client environments that the server can access during a specific round for training with the capability to extrapolate to new and unseen environments in server/client side.

3.2. How can a federated setting help causal invariance learning?

In the invariance-based causal learning approaches to domain generalization, we always aim to learn a classifier that only uses invariant features which are the features whose distribution is constant across environments. In such cases the question on how many environments are needed to identify the causal features remains the most complicated one to answer. In (106), it has been shown that popular approaches like IRM(77) can require a large number of environments to work which is linear in the number of spurious features, even on a simple linear data model. In order to achieve invariance across all participating client environments, IRM (10) (115) requires sufficient diversity across environments and assumes that for a set of training environments, $\mathcal{E}_{train}$ lie in linear general position of degree r, if and only if for some $r \in \mathbb{N} : |\mathcal{E}_{train}| > d - r + \frac{d}{r}$. Also for all non-zero $x \in \mathbb{R}^d$:

$$\dim\left(\operatorname{span}\left(\left\{\mathbb{E}_{X^c}\left[X^c X^{cT}\right]x - \mathbb{E}_{X^c,\epsilon^c}\left[X^c \epsilon^c\right]\right\}_{c\in\mathcal{E}_{train}}\right)\right) > d - r$$

The number of environments is unrealistic in practice for a centralized learning approach employed for causal learning. This is exactly where federated learning setting fits in for causal learning. Federated setting has exactly the required practical setting of non identical and highly diverse client train environments to help achieve invariance.

3.3. How can causal invariance learning be of help in a federated learning setting?

As compared to associational models that are being used in federated learning, models that are learnt wrt causal structure always exhibit better generalization to Non IID data ie data from different distributions.

One of the main attacks posed to federated learning is membership inference attacks which is the case wherein only the model predictions could be observed by the attacker (107), (87).

In (87), it has been proved that the distribution of the training data as well as the generalizability of the model significantly contribute to the membership leakage. Particularly, they show that overfitted models are more susceptible to membership inference attacks than generalized models. Hence it could be inferred that such inference attacks could be nullified to some extent with learning networks that exibit better generalization.

In (116), the generalization property of causal learning has been proved wherein they establish a theoretical link between causality and privacy.It is shown that models learnt using causal structure generalize better to unseen data, especially on data from different distributions than the train distribution. It was also proved that causal models provide much better differential privacy guarantees as compared to the current associational models that we use.

In (53), it has been proved that a causal view to domain adaptation with multiple source domains generalizes better and noted that the background causal knowledge in the data-generating process helps greatly in domain adaptation. In each situation, it is beneficial to investigate what knowledge is appropriate to transfer and find the optimal target-domain hypothesis. This gives an intuitive interpretation of the assumptions underlying certain previous methods and motivates new ones. Under appropriate assumptions, the availability of multiple source domains allows a natural way to reconstruct the conditional distribution on the target domain; (53) proposed to model $P_{X|Y}$ (the process to generate effect X from cause Y) on the target domain as a linear mixture of those on source domains, and estimate all involved parameters by matching the target-domain feature distribution.

Instead of enhancing just privacy preservation, we focus on both privacy and the robustness of federated learning so that the global model should behave correctly even when there is a large portion of malicious participants.

3.4. Federated Learning On Non IID Data

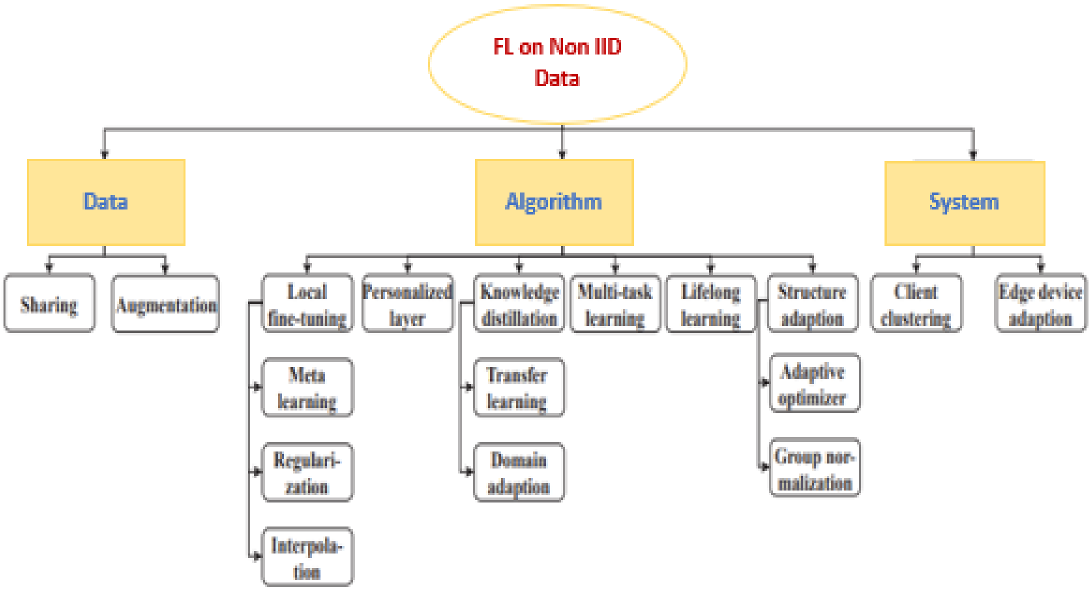


Fig. 3.2. Existing approaches in FL to handle Non IID data (139).

In a federated learning setting, as the client local data distributions mostly differ from the global data distribution, the averaged local model parameters can exhibit high divergence from the global optima (136), particularly when the number of epochs for local updates is large (127)(109)(31). Federated averaging (81) shows high degradation in performance on Non-IID / heterogeneous setting (136) as client models might end up converging to different models because of the heterogeneity in local data distributions leading to high divergence between the calculated global model and the ideal/expected model.

As depicted in 3.2, although several approaches(139) have been experimented based on data, algorithm and system to handle distributional shift in client/server side data, causal learning techniques have not yet been included in such studies. With the evident benefits of causal machine learning in out of distribution generalization, this thesis studies some approaches to handle some of the existing issues in federated learning with causal learning.
The approaches we propose mainly focus on the below specified strategies:

3.4.1. Data Sharing Strategies

Data sharing techniques have been very well explored in the field of federated learning for dealing with heterogeneous/ Non-IID datasets. This was introduced initially in (136) where the concept of a globally shared dataset G was first used. This global dataset G, which is stored on the server, has a uniform distribution and a random percentage of G is transferred to all participating clients in each round of the training process. Each participating client updates their respective local model by training on both local data pertaining to that particular client as well as the shared global data G. This widely used approach contributed to enhancing the test accuracy on CIFAR10 dataset by almost 30% with a small portion of globally shared data.
Many similar approaches have been proposed in literature to enhance the model performance on Non-IID data in a federated setting, some specifically propose sharing local data with server (133) (122).

3.4.2. Knowledge Distillation

The idea of information transfer from large to small models(16) forms the base of knowledge distillation (48)(43)(16). This concept has been widely in use in a federated learning setting to transfer knowledge from server to client, client to client/server or clients to a particular client to improve the model performance on unknown heterogeneous data. Federated Transfer learning(69) (126) plays the most significant role in enabling knowledge distiallation in a federated setting.
(68) proposed an ensemble distillation strategy that enhances robust fusion of multiple client models which proved to have evident benefits in alleviating risk of leakage. (23) developed a collaborative robust learning method for uploading learned features from clients instead of

local models to implement client level personalization. To further enhance personalization and generalization to unseen mobile devices, (126) developed a next-word prediction model with different hyperparameters. Many of these knowledge transfer techniques rely on homomorphic encryption (41) and secret sharing (24) protocols to enhance privacy. Another approach to decentralize federated learning was proposed by (65) which involved mutual knowledge transfer among clients in a peer-to-peer manner without any involvement of a central server. (65) stated that the concept of mutual knowledge transfer is very important in federated learning domain as it helps mitigate the influence of label-shift (135).

3.4.3. Domain Adaptation

Domain adaptation is another significant aspect of knowledge distillation where the emphasis is on eliminating the differences between the data shards among the clients. A domain adaptation algorithm called federated adversarial domain adaptation presented in (94) aims to solve the domain shift problem using adversarial techniques. Another instance is the FeMD algorithm proposed in (66) which enables clients to train their models on local data. This is acheived by transfering knowledge from a public dataset without privacy leakage risk by training the clients' models with this public data and then another round of training is done using the private local dataset. This tackles the privacy leakage risk.

3.5. Proposed Causal Federated Learning Approaches

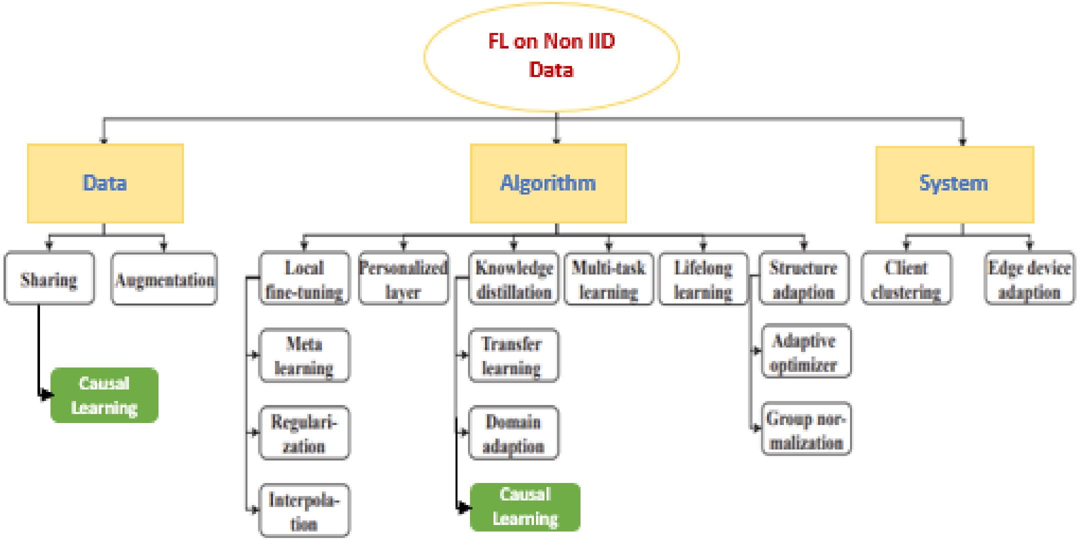


Fig. 3.3. Existing approaches in FL to handle Non IID data with our proposed approach highlighted in green.

With plethora of work on feature transfer among clients and server in a federated learning setting to enhance robustness, applying causal risk minimization strategies to these techniques can further help enhance the generalization of the final learned model to unseen test clients. Although local data sharing methods outperform global data sharing strategies by a big margin when it comes to Non-IID setting, this has very evident privacy issues attached to it. While global shared dataset seems to be a convincing strategy to deal with data heterogenity keeping the client data private, it is hard to attain a uniformly distributed global dataset, provided the fact that the server can never be well informed about the data distributions present in the participating client datasets. However, with our proposed approach of learning causally invariant features common to both the client and global shared data, this issue can be solved.

3.5.1. Approach 1 - CausalFed

:

Keeping the data private, coming up with a way to collaboratively learn causal features common to all the participants was a bit challenging. In our federated causal learning framework, the inital layer (local) is the one where in each of the participating client entities do the local training for extracting features from their respective input data and outputs the respective features in the form of numerical vectors. Consider client data $\mathcal{D}_C = (x_i^C, y_i^C)_{i=1}^{N_C}$ where x_i^C is i^{th} input and y_i^C is i^{th} label for client C. The hidden representation of each participating client is produced by neural network as

$$h_i^C = \phi^C(x_i^C)$$

where $h^C \in \mathcal{R}^{N_C \times d}$, d is the dimension of hidden representation layer. The federation layer is for the participating clients to exchange intermediate training components and train the federated model in collaboration by minimizing the empirical average loss as well as regularizing the model by the gradient norm of the loss for both the participating entities/environments as (77):

$$\sum_{C,i}^{S,N_C} \mathcal{L}_d(w \circ h_i, y_i) + \lambda \sum_{C}^{S} \left\| \nabla_{w|w=1.0} \sum_{i}^{N_C} \mathcal{L}_d(w \circ h_i, y_i) \right\|^2$$

where S equals set of clients/ source domains, N_C equals number of samples per client C, $\mathcal{L}_d$ equals classification loss, and h, y to represent the hidden representation and its corresponding true class label and λ is hyperparameter.The error is the usual error we would use for any machine learning problem calculated on each environment.

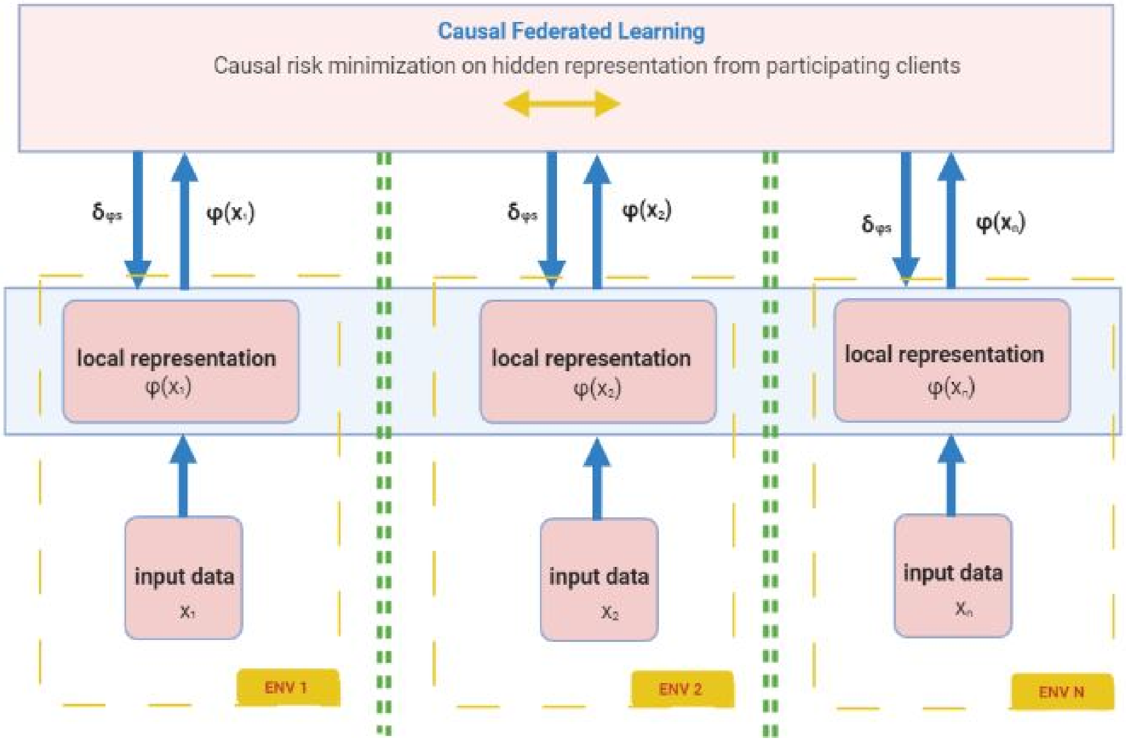

Causal Federated Learning
Causal risk minimization on hidden representation from participating clients
$\delta_{\varphi s}$
$\varphi(x_1)$
$\delta_{\varphi s}$
$\varphi(x_2)$
$\delta_{\varphi s}$
$\varphi(x_n)$
local representation $\varphi(x_1)$
local representation $\varphi(x_2)$
local representation $\varphi(x_n)$
input data x_1
input data x_2
input data x_n
ENV 1
ENV 2
ENV N

Fig. 3.4. Causal Federated Learning

It how well the model is performing in each environment whereas the penalty term measures how much the performance could be improved in each client environment with one gradient step as well as punishes high gradients (for example a case in which a large improvement in one particular participating client environment would be possible with one more epoch of learning) which in turn accounts for the model having a low gradient in each participating client environment ensuring that the learning is balanced between all the client environments. This is perspective on the domain-invariance of representation for domain generalization where we do not seek to match the representation distribution of all client domains, but enforce the optimal classifier on top of the representation space to be the same across all participating client domains. The intuition is that the ideal representation for prediction is the cause of y, and the causal mechanism should not be affected by other factors, thus is domain-invariant. With Invariant Risk Minimization (IRM) (77) we attempt to learn invariant predictors in a federated learning setup that can attain an optimal empirical risk on all the participating client domains.

Algorithm 1 CausalFed

```
            **ServerCausalUpdate:**
  Initialize $\mathbf{W}_0^s$
  **for** each server epoch, t = 1,2,..k  **do**
    Select random set of S clients
    Share initial model with the selected clients
    **for** each client $k \in S$ **do**
      $(\boldsymbol{\phi}(x_t^k), \mathbf{Y}^k) \leftarrow ClientRepresentation(k,\mathbf{W}_t^k)$
      Evaluate loss $\mathcal{L}_k$
    **end for**
    $\mathcal{L}_s = \sum_k^S \mathcal{L}_k + \lambda \sum_k^S \left\| \nabla \mathcal{L}_k \right\|^2$
    $\mathbf{W}_{t+1}^s \leftarrow \mathbf{W}_t^s - \eta \nabla \mathcal{L}_s$
  **end for**
  $\mathbf{W}_t^k \leftarrow ClientUpdate(\nabla \mathcal{L}_s)$
        **ClientRepresentation($\mathbf{W}_t^k$):**
  **if** k is first client to start training **then**
    $\mathbf{W}_t^k \leftarrow$ initial weights from server
  **else**
    $\mathbf{W}_t^k \leftarrow \mathbf{W}_{t-1}^{k-1}$from the previous $ClientUpdate(\nabla \mathcal{L}_s)$
  **end if**
  **for** each local client epoch, i=1,2,..k **do**
    Calculate hidden representation $\boldsymbol{\phi}(x_t^k)$
  **end for**
  **return** $\boldsymbol{\phi}(x_t^k)$ and $\mathbf{Y^k}$ to server
              **ClientUpdate:**
  **for** each client $k \in S$ **do**
    $\mathbf{W}_{t+1}^k \leftarrow \mathbf{W}_t^k - \eta \nabla \mathcal{L}_s$
  **end for**
  **return** $\mathbf{W}_{t+1}^k$ to server
```

3.5.2. Approach 2 - CausalFedGSD

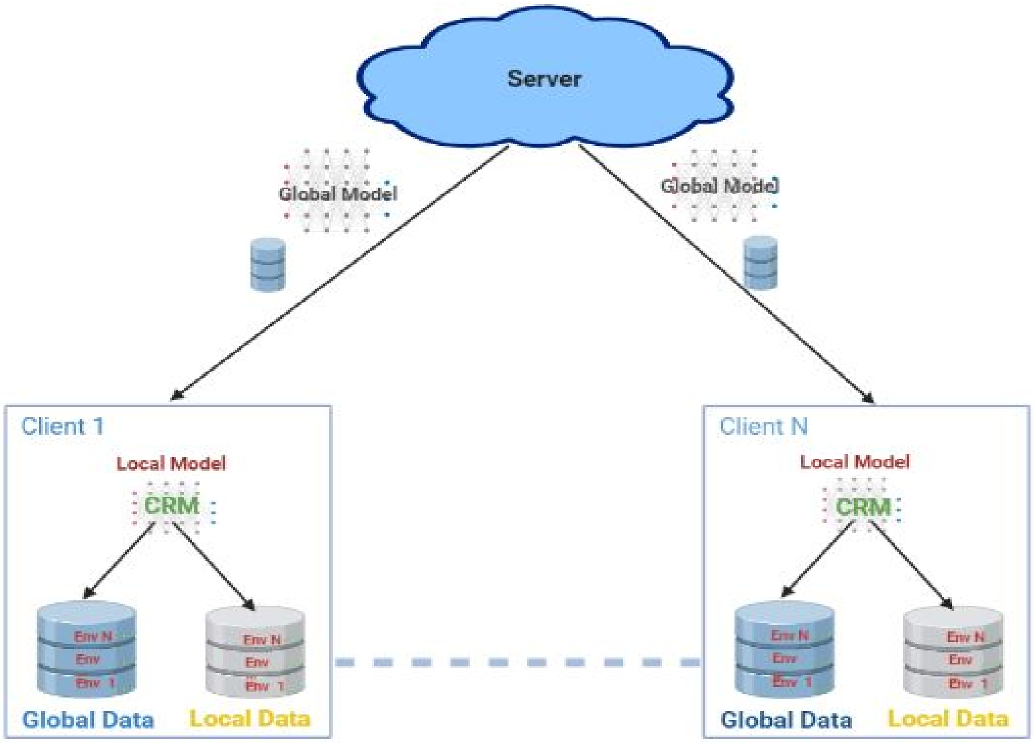


Fig. 3.5. Global Data sharing strategy

With our previous approach, there are many privacy concerns regarding the client data which are still not addressed due to which depending on a global data set(with different enviroments) to enhance causal feature learning within a federated learning setup seems plausible. As we have no control on the clients' data, we can distribute a small subset of global data containing a distribution over all the classes/enviroments from the cloud to the clients during the initialization stage of federated learning.

The local model of each client is learned by minimizing the empirical average loss as well as regularizing the model by the gradient norm of the loss for both the shared data from server (Global Environment) and private data from each client (Local Environment). This enhances the learning of causal/invariant features common to both the client and global data environments without losing the privacy of client side data.

In (134), it has been shown that globally shared data can reduce EMD (earth mover's distance) between the data distribution on clients and the population distribution which can help in improved test accuracy. As this globally shared data is a separate dataset from that of the client, this approach is not privacy sensitive with respect to clients. But it poses privacy issues to the global server. One possible approach to tackle this issue would be to anonymize data or de-identify data. This involves removal of sensitive/personally idenitifiable information pertaining to the global data subject so that sample anonymity is preserved (90).

Algorithm 2 CausalFedGSD

```
                **ServerUpdate:**
G $\leftarrow$ 5% distribution over all environments present in server
Initialize $\mathbf{W}_0$
Initialize random portion of G as $G_0$
**for** each server epoch, t = 1,2,..k **do**
    Select random set of S clients
    Share $G_0$ and initial model with the selected clients
    **for** each client $k \in S$ **do**
        $\mathbf{W}_{t+1}^k = ClientUpdate(k,\mathbf{W}_t)$
    **end for**
    $\mathbf{W}_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} \mathbf{W}_{t+1}^k$
**end for**
                **ClientUpdate(W):**
$\mathcal{E}_{\mathrm{tr}} \in$ [Client Env] $\bigcup$ [Global Env]
**for** each local client epoch, t=1,2,..k **do**
    $L_{\mathrm{IRM}}(\Phi, \mathbf{W}_t^k) = \sum_{e \in \mathcal{E}_{\mathrm{tr}}} R^e(\mathbf{W} \circ \Phi) + \lambda \cdot \mathbb{D}(\mathbf{W}, \Phi, e)$
    $\mathbf{W}_t^k = \mathbf{W}_t^k - \eta \nabla L_{\mathrm{IRM}}(\mathbf{W}_t^k)$
**end for**
return W to server
```

3.6. Implementation Details

3.6.1. Dataset Details

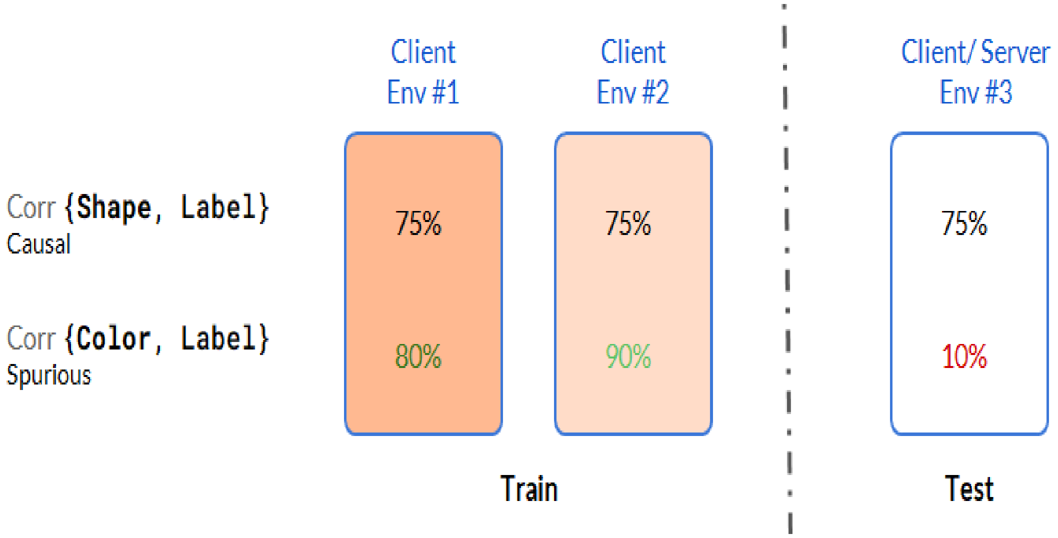


Fig. 3.6. ColoredMNIST

Colored MNIST: Unlike the MNIST dataset which consists of digits 0-9 in grayscale, the colored MNIST dataset consists of input images with digits 0-4 colored red and labelled 0 while digits 5-9 are colored green with label with shape of the digit as the causal feature. In our causal federated learning setup, we split the dataset to two environments, each corresponding to a participant/client. We sample 2000 data points per client/server domain. Within the client environments, 80 - 90 % of inputs have their color correlated to the digit whereas within the central server test enviroment has just 10% color-digit correlation which helps in testing the robustness despite the spurious correlation within the inputs.
Rotated MNIST: This dataset consist of original MNIST split to multiple client/participating environments by rotating each digit[0-9] with angles $0°, 15°, 30°, 45°,$ and $60°$. We sample 1000 data points per client/server environment. The server side test domain consist of digits with angles 75° and 90°
Rotated Fashion MNIST: Fashion-MNIST is a dataset of Zalando's article images — consisting of a training set of 60,000 examples and a test set of 10,000 examples. Here again we split the dataset to multiple client/participating environments by rotating each digit[0-9] with angles $0°, 15°, 30°, 45°,$ and $60°$. We sample 10000 data points per client/server environment. The server side test domain consist of digits with angles 75° and 90°.

3.6.2. Attack Implementation

Backdoor Attack Setting
For an initial basic analysis, we tried two backdoor attacks:

- A single-pixel attack, where in the attacker changes the top-left pixel color of all the inputs, and mislabels them.
- A semantic backdoor where in the attacker selects certain features as the backdoors and misclassifies them. For example, the attacker classifies digits rotated 15° with label 7 as 0.

In both the cases, CausalFed exhibited better resilience as compared to FedAvg.

Membership Inference Attack Setting We use pytorch code provided by M. Nasr, R. Shokri(102) For this particular attack, we consider a scenario wherein stochastic gradient descent is used to train the target model and we do one global aggregation per training epoch which implies that all the participating clients will be reporting their local models to the central server after each step of local gradient descent followed by the server aggregating the models by taking a weighted average which(resulting model) in turn is sent to the participating clients. This setup is considered to give a target model which can be considered equivalent to the model trained by gradient descent on the union of all the participating client train data.

The main idea is that each training data point affects the gradients of the loss function such that the adversary can use Stochastic Gradient Descent algorithm (SGD) to extract information from other clients' data (91). The adversary can perform gradient ascent on a target data point before local parameter update. SGD reduces the gradient,in case the considered data point is part of a client's set resulting in a succesful membership inference. Attack can come from both the client side and the server side. An adversarial client can observe the aggregated model updates and extract information about the union of the training dataset of all other participants by injecting adversarial model updates. For a server side attack, it can control the view of each target participant on the aggregated model updates and extract information from its dataset(91).

In our implementation, we sample 1,000 datapoints for Rotated-MNIST, 500 datapoints for Colored-MNIST and 1,000 datapoints for Fashion-MNIST from the original train and test dataset to create the attack-train and attack-test dataset. We use pytorch code provided by (102)(91)

Property Inference Attack Setting In this setting, the adversary aims to recognize patterns within a model to reveal some property which the participant client never intended to disclose. Melis et al(49), has listed different approaches on how to infer properties of participant client train data whose features have no correlation with the ones that characterize the classes of the model.

The main idea behind this attack is that, at each round, each client's contribution is based on a batch of their local training data, so the attacker can infer properties that characterize the target dataset for which the adversary needs sample train data, which is labeled with the attribute to be infered.(20) It is aimed at infering properties of client data that are uncorrelated with the features that characterize the classes of the model. In our experiments we decided on client domain as the attribute which is to be inferred by the adversary. Another such attribute that is uncorrelated with the final prediction is the color of the input.

We observe that federated causal models provide better pivacy guarantees(116) against this attack which could be owed to the fact that inversion based on learning correlations between attributes and final prediction, e.g., using color to predict the digit, can be eliminated by causal models, since a non-causal feature will not be included in the our final causal federated model.

3.7. Results

In our experiments, we compare the performance of federated averaging (Fed-Avg) with the following approaches:

Fed-ERM Within the CausalFed setup, this approach minimizes the empirical average of loss over training data points and treats the data from different domains as i.i.d. ERM loss is given by:

$$\sum_{C,i}^{S,N_C} \mathcal{L}_s(w \circ h_i,y_i)$$

where S equals set of clients/ source domains, N_C equals number of samples per client C, $\mathcal{L}_s$ equals classification loss.

CausalFed-RM In this approach, we minimize the random match(RMatch) causal loss (76) within the CausalFed setup. RMatch loss is given by:

$$\sum_{C,i}^{S,N_C} \mathcal{L}_s(w \circ h_i,y_i) + \lambda * \sum_{\Omega(j,k)=1 | j \sim N_C, k \sim N_{C'}} \text{Dist}(h_j, h_k)$$

where Ω represents the match function used to randomly pair the data points across the different client domains.

CausalFed-IRM In this approach, we minimize the IRM loss (77) within the CausalFed setup.

3.7.1. Evaluation Setup

In our setting, client nodes and server nodes are run on separate computing nodes of compute canada cluster. All clients are bound to update the model in each epoch of training. Approximately 12 GB RAM was allocated per slurm job. Python 3.7 is the language used for all the code implementation. Pytorch 1.10 is the machine learning library employed for all the prototypes. The network architecture and learning rates are selected based on model performance during intial rounds of training.

Tableau 3.1. Network Architecture

Architecture	No of Layers	Kernel spec	Learning Rate
LeNet	5	(5x5), (2x2)	0.003
AlexNet	8	(11x11), (5x5), (3x3)	0.0001
ResNet18	18	(7x7), (3x3)	0.0004

Federated Averaging denoted as FedAvg is considered the benchmark in our comparison study. The results shown are observed on 150 global epochs on a set of 10 clients with batch size set to 512. To test the variation of accuracy over each global epoch, we computed CV which is the coefficient of variation to measure the dispersion. It is given by the ratio of standard deviation to mean. Based on our analysis, CV coefficient values range from 0.02 - 3.16 for train results and 0.3 - 4.8 on test results after epoch 5.

3.7.2. Approach 1 - CausalFed

We perform our evaluation to analyze the relation between learning causal features and OOD generalization accuracy gap within client-sever in a federated learning setting. We observed that when clients have out of distribution data in a federated setting, FedAvg as well as FedERM does not fare well in the test data set though they give highly accurate train results whereas CausalFed-RM and CausalFed-IRM perfoms much better on test data.

Tableau 3.2. Comparison of methods in terms of training accuracy (mean ± std deviation)

Dataset	Arch	Fed-Avg	Fed-ERM	CausalFed-RM	CausalFed-IRM
Colored MNIST	ResNet18	80.3±.18	**82.97** ±.2	60.42 ±2.9	59.33 ±3.2
Rotated MNIST	ResNet18	85.2±.14	**86.5** ±.12	79.8 ±1.7	80.2±2.1
Rotated FMNIST	LeNet	81.4±.12	**82.3** ±.16	72.1 ±1.4	71.5 ±2.6

Tableau 3.3. Comparison of methods in terms of testing accuracy (mean ± std deviation)

Dataset	Arch	Fed-Avg	Fed-ERM	CausalFed-RM	CausalFed-IRM
Colored MNIST	ResNet18	11±.17	10.2±.8	**65.62** ±3.1	60.3 ±4.8
Rotated MNIST	ResNet18	82.7±.5	82.9±.7	**90.2**±2.9	89.1±4.1
Rotated FMNIST	LeNet	69±.2	71.6±.16	**74.6**±3.9	73.9±4.6

We perform our evaluation to compare privacy attack accuracy of FedAvg with CausalFed training methods on Colored-MNIST, Rotated-MNIST and Fashion-MNIST datasets. The privacy leakage on each of the attacks is measured by testing the accuracy of attack model.

Tableau 3.4. Leakage on inference attack

Dataset	Fed-Avg	Fed-ERM	CausalFed-RM	CausalFed-IRM
Colored MNIST	79.21 %	79.45 %	58.57 %	56.9 %
Rotated MNIST	84.4 %	85.24 %	68.3 %	64.4 %
Rotated FMNIST	76.61 %	78.23 %	57.55 %	55.7 %

3.7.3. Approach 2 - CausalFedGSD

Tableau 3.5. Comparison of methods in terms of training accuracy (mean ± std deviation)

Dataset	Arch	Fed-Avg	Fed-ERM	CausalFedGSD-RM	CausalFedGSD-IRM
CMNIST	ResNet18	80.3±.15	**82.97**±.4	57.42±2.1	55.32±2.9
RMNIST	ResNet18	85.2±.12	**86.5**±.21	73.7±2.2	77.2±2.9
RFMNIST	LeNet	81.4±..3	**82.3**±.21	69.2±3.5	68.6 ±4.2

Tableau 3.6. Comparison of methods in terms of testing accuracy (mean ± std deviation)

Dataset	Arch	Fed-Avg	Fed-ERM	CausalFedGSD-RM	CausalFedGSD-IRM
CMNIST	ResNet18	11±.2	10.2±.4	**55.62**±2.5	52.3±4.6
RMNIST	ResNet18	82.7±.19	82.9±.15	**85.2**±2.8	83.1±4.5
RFMNIST	LeNet	69±.2	71.6±.18	**71.9**±3.7	70.2±4.8

We observed that when clients have out of distribution data in a federated setup, FedAvg as well as FedERM does not fare well in the server side test data set though they give highly accurate results on train data(iid) whereas CausalFed-RM and CausalFed-IRM perfoms much better on test data(non iid). With respect to privacy, we observe that in our setup with an out of distribution(OOD) test set, the membership inference attack accuracy of a federated causal client adversary model is much lesser as compared to a federated setup with associational client models. It was also observed that federated causal models provide better pivacy guarantees against property inference attacks which could be owed to the fact that inversion based on learning correlations between attributes and final prediction, e.g., using color to predict the digit, can be eliminated by causal models, since a non-causal feature will not be included in our final causal federated model.

3.8. Some Possible Steps To Further Enhance Privacy

Homomorphic Encryption: A cryptographic technique that preserves the ability to perform mathematical operations on data as if it was unencrypted i.e., plain text. For example, performing neural network computations on encrypted data without the need of first decrypting it. Homomorphic encryption is studied for private federated logistic regression(88) Moving forward, these techniques can be applied to CausalFed approach.

Secure Multiparty Computation: The technique is based on splitting data among collaborating entities to perform joint computation but prevents any collaborating entity from gaining knowledge of the data. For example, identifying the common patients among two hospitals without disclosing the respective hospital patient's list. So many examples worth trying in literature (25)

Differential Privacy The alteration of a dataset to obfuscate individual data points while retaining the ability of interaction with a data within a certain scope (privacy budget) and of statistical analysis. The approach can also be applied to algorithms. For example, randomization of data to omit relationships between individuals and respective data entries. It provides privacy preservation against membership-inference attack in the model inference stage (103)

Data Anonymization Eventhough we proposed anonymization as a solution to privacy leakage in CausalFedGSD, data anonymisation has to balance well between privacy-guarantee and utility as removal of sensitive attributes might result in degradation of the utility of the dataset. Also it is possible for an adversary to combine the global shared data with other anonymous datasets in an attempt to compare and re-identify the original data subject. This kind of an attack is termed linkage attack (39) and it is one of the most prominent attacks for de-anonymization. To prevent the occurence of linkage attack in CausalFedGSD setting, we will look into exisitng techniques proposed to handle this attack such as k-anonymity (62), l-diversity (73) and t-closeness - a technique built on both k-anonymity and l-diversity that preserves the distribution of sensitive attributes in a dataset so that it reduces the risk of re-identifying a data subject in a same quasi-identifier group(64).

3.9. Challenges Posed To Proposed Approaches

Research work on incorporating causal learning techniques into federated learning framework is crucial. However, we still have challenges to make this approach work in practical scenarios. Some of the main challenges posed are:

- We need to develop better schemes to learn the causal knowledge in a way that it can well capture the invariant features common to all participating clients.In contrast to the current sequential and centralized causal learning techniques where the causal/invariant features is mostly represented in one universal pre-trained model, we have it distributed among local models. In our case, each participating client has significant control over building its local model. We should strive to acquire a balance between autonomy and generalization performance of the causal federated learning models.
- We need to come up with distributed learning approaches that would ensure preserving the privacy of the shared causal representation of all participating clients. Under the federated learning framework, transfer knowledge is not only learned in a distributed manner, but also is typically not allowed to be exposed to any participant. Thus, we need to figure out precisely what each participant contributes to the shared representation in the federation and consider how to preserve the privacy of the shared representation.
- Downloading parts of global dataset to each client for model training in CausalFedGSD setting can be considered as a violation to the requirement of privacy preserving learning in Federated setting. Moving forward, we will study possible approaches to tackle this issue.

3.10. Conclusion

In this work, we show that CausalFed is more accurate than non-privacy-preserving approaches as well as superior to non-federated associational learning approaches in comparison to existing privacy enhancing approaches in federated setup which suffer from pretty high accuracy loss. We were able to prove that causal feature learning can enhance out of distribution robustness in federated learning. Moving forward, we need to analyse the performance of this approach in real world datsets as well as compare various other causal learning approaches which can further enhance the out of distribution robustess and improve leakage protection in our current setup. We believe that CausalFed is a general approach that offers several extensions for future work.

Chapitre 4

Federated Incentivization With Blockchain

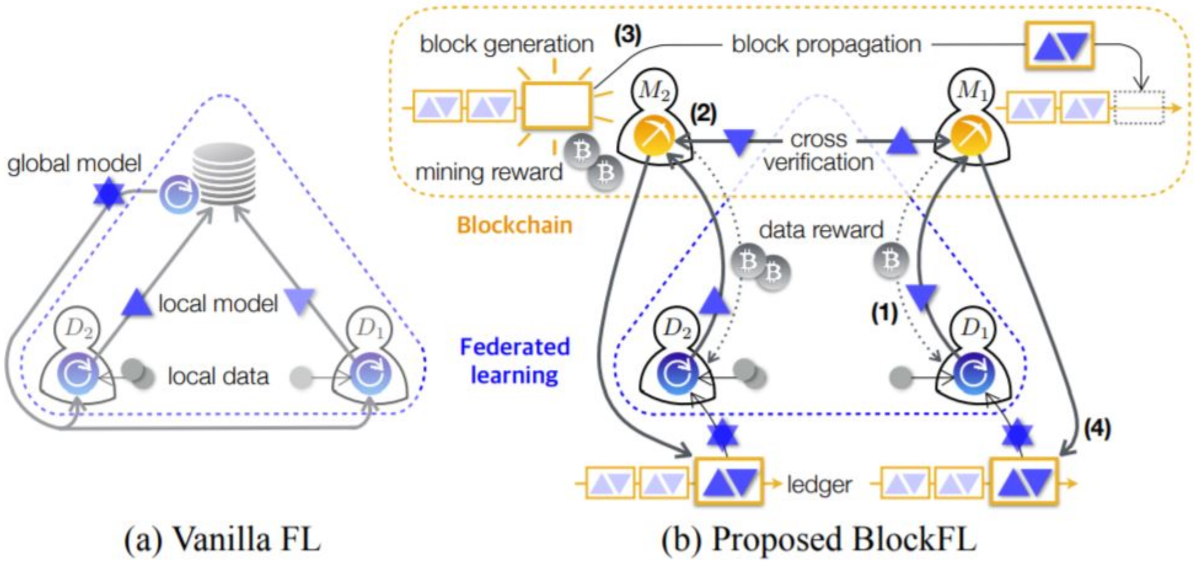
This chapter partly consists of a reproduction of (78) 978-1-7281-2542-8/19/$31.00 @2019 IEEE DOI 10.1109/ CyberC. 2019.00018. This was accepted to International Conference on Distributed Computing and Knowledge Discovery with reviewers rating this an innovative approach to incentivize participants in a federated learning setting.
The idea was initially conceptualized by the author of this thesis. She is the joint first author of this paper with Ismael Martinez. She came up with the idea, wrote the entire python section of the code for both blockchain and federated learning training phase and formulated the class validation error metric. Both the authors contributed equally to writing. This chapter is included in this thesis as an introduction to the next chapter which is aimed at incentivizing participants in the causalfed (37) setting.

4.1. Background

In today's data market, users generate data in various forms including social media behaviour, purchasing patterns, and health care records, which is then collected by firms and used either for sale or for in-house data analytics and machine learning . As a result, each of us is essentially giving away a personal resource for no reward. In addition, these organizations have full access to our data, which can be a major invasion of privacy depending on the type of data collected. One proposed method of mitigating this issue of ownership and privacy when the purpose of the data is proprietary machine learning is *Federated Learning* (57) (83), where an owner sends the training model to users who train on their local data and send back only the updated weights of the model. By doing this, a user never unveils the data to the owner, and keeps ownership of said data. A secondary result of this type of training is that users with sensitive data such as health care data are more likely to partake in the training, meaning the owner also receives more data to use for training.

There still remains the concern of handing out our data, a useful resource to organizational training models, for free. We propose the use of blockchain to facilitate the uploading and tracking of updates from users, as well as rewarding users for the data they used in computation. An additional benefit to using a blockchain is that it renders the updates immutable and thus secure. The combination of data privacy and security coupled with rewards for uploads renders this system desirable to a larger scope of users, some with sensitive data, allowing organizers to collect a larger pool of data from a wider set of users.

BlockFL (56) uses blockchain to reward users for their local updates proportional to how many local data points are used as shown in Figure 4.1. The payment to devices is left to the miner to pay "out-of-pocket", which is not a lasting solution if miners pay devices more than they are rewarded for blocks.This device reward benefits an honest node; however, this value may be inflated by a malicious node seeking higer reward.



(a) Vanilla FL

(b) Proposed BlockFL

Fig. 4.1. As described in the proposal for BlockFL (56), the architecture of BlockFL compared to "Vanilla" Federated Learning(83) @2019 IEEE.

Kurtulmus and Daniel (59) proposed a blockchain implementation of machine learning to reward the user who could produce a valuable machine learning model for a publicly available dataset and evaluation function published by an organizer. One large problem that arises with this system is that all model evaluations are done on the blockchain which yields large gas costs; many users must each pay gas for their models to be evaluated, however only a small selected group is paid out.

DeepChain looks into using blockchain for Distributed Deep Learning applications (128) as a means to keep both data and model private and secure. DeepChain proposes an incentive-based blockchain mechanism where both *parties* – those who upload data to the model, and *workers* – the ones who process the data and update the model, get paid an amount π_P and π_W based on their *contribution* ω_P and ω_W for parties and worker respectively. In addition, DeepChain proposes the use of a *penalty mechanism* whereby rewards to dishonest nodes are frozen and re-allocated. Although these two mechanisms works for a DML system where it is public how many data points each party is uploading, this cannot be applied to Federated Learning since only the update values δ are uploaded. Regarding a penalty mechanism, there is no need for one if Smart Contracts (19) are used. With a Smart Contract validating and administering the payment, a faulty transaction would fail and the payment would not be administered. The consensus protocol used by DeepChain is *blockwise-BA*, which elects a randomly chosen using cryptographic sortition (86) and a seed that changes with every block. Once a worker is selected, the worker creates a block, which is verified by a selected committee before being added to the blockchain. This method relies on choosing an honest committee, and for the random algorithm to be negligibly close to perfectly random, both issues which may not be true in practice.

One blockchain implementation of ML sought reward the user who could produce the best ML model for a publicly available dataset and evaluation function published by an organiser (59) in an architecture and process akin akin to Kaggle Data Science Competitions (1). In this workflow, users would produce and train an ML model with the published dataset that would maximize the score given when the model is applied to the evaluation function; either the first model, the best model, or both would be rewarded by means of smart contracts. One issue addressed is that of a biased test/train split by the organizer that could cause good models to not be rewarded due to poor performance on the chosen split; the proposed solution was to randomize the split in a fashion out of the organizer's control, however there is nothing stopping the organizer for adding new data to the system. One large problem that arises with this system is that all model evaluations are done on the blockchain. Although the issue of too many models being evaluated at once is addressed, the larger issue of the large gas costs associated with the evaluation is still present; many users must each pay gas for their models to be evaluated, however only a selected group is paid out. One consideration would be to have this computation and evaluation done off chain, with only the results returned and recorded in the blockchain.

Another study (84) looks into Distributed Deep Learning in order to take advantage of increased processing power and big data; it is proposed to use blockchain to create an incentive-compatible data market. Similarly, (29) attempts to make a DML system with user rewards based on Ethereum Smart Contracts (129). It also briefly mentions a system of data owners granting access to others to view its data (29).When we are working with blockchains that are not fully public, it is proposed to be using a form of Access Control (74) (75) (123). One proposed method is to use *Attribute-Based Access Control* whenever working with blockchains (74)(33) (49). Another proposed form of Access Control in blockchain is through the use of Smart Contracts (74) (75) (123) (19) (21).

Computation of model updates is proposed in Federated Learning (57) (83) to either be distributed over a higher number of smartphones, or to run complex computations in idle states. Both of these methods are based on the assumption that a smartphone has the processing capabilities for these computations in their idle states.

A number of sites have appeared with services around a decentralized data marketplace. The Ocean Protocol (2) is a blockchain service that describes itself as "a tokenized service layer that exposes data, storage, compute and algorithms for consumption". It is a data marketplace where users can sell or buy data in a secure, safe and transparent fashion, and was created with data sharing for AI in mind. Per the White Paper (93), this service leverages the Ethereum interface for smart contracts and token exchanges. Another such marketplace is Wibson (3) where in addition to transparency and anonimity, ensures users maintain control of the use of their data after it is sold. The price of data is dictated by the market, and Wibson utility tokens are rewarded to facilitate the use of Wibson on top of the Ether that's rewarded for the data; the system is built on Ethereum and the tokens are stated as ERC20 tokens (79). The Datum Network (4) uses DAT tokens to be used within the blockchain network with these tokens available for purchase or sale from Ether (104); there is however the problem of value, since the price at which a user is selling their data is unclear. The Datum Network does propose a functionally rich system where data is able to be queried and searched (104).

One key factor that isn't mentioned in many studies is the format of distributed data (128) (84) (29) (57). (83) (56); we may not be able to assume homogeneneous data with heterogeneous devices. In addition, a standard for the format of δ uploaded updates is not well defined from the device to the miner (57) (83) (56).
The limitations of the related work can be summarized as inaccuracy or inefficiency in rewarding user contributions, and lack of scalability of data on the blockchain.

4.2. Record And Reward Federated Learning Contributions With Blockchain

The main contributions of this chapter can be summarized as follows:

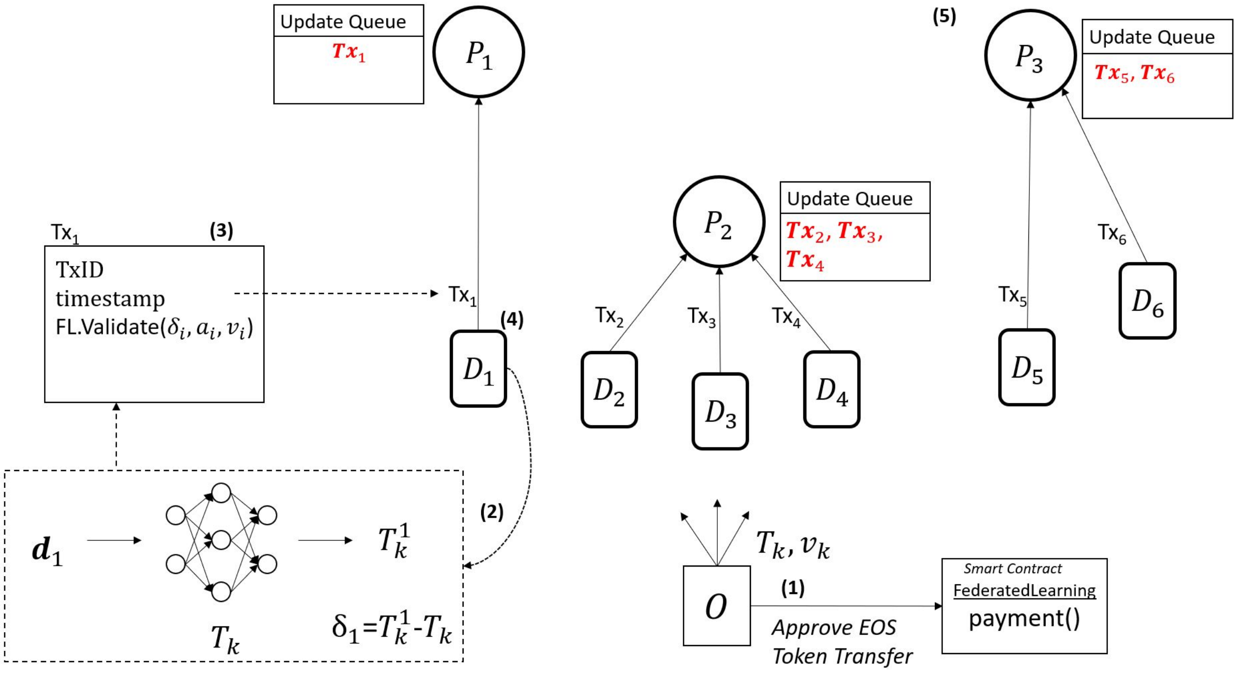


Fig. 4.2. The workflow of calculating and uploading update values δ for validation,as explained in Section 4.3.3 @2019 IEEE.

- Merging Federated Learning with blockchain to ensure both data privacy (50) (82) and security, and thus motivate more user contributions.
- Using EOS Blockchain and IPFS to *record* uploaded updates in a sclalable manner and *reward* users based on training data cost.
- Proposal of a Class-Sampled Validation Error Scheme (CSVES) for validating and rewarding only valuable uploaded updates via Smart Contracts.
- Simple implementation with Python and Hyperledger Fabric to verify the feasibility of the system, with plans to implement a PoC in EOS at a later date.

4.3. Proposed Design And Architecture

Taking related work into consideration, we choose to make adjustments to improve privacy, access control and storage; the architecture and workflow of the proposed system are shown in Fig. 5.3 and 5.4.

The following assumptions are made regarding the devices and data in the system.

- A smartphone device has enough storage to store the kth model.
- A smartphone device does not necessarily have enough extra storage to store the entire blockchain.
- The training data for the model is homogeneous across different devices.

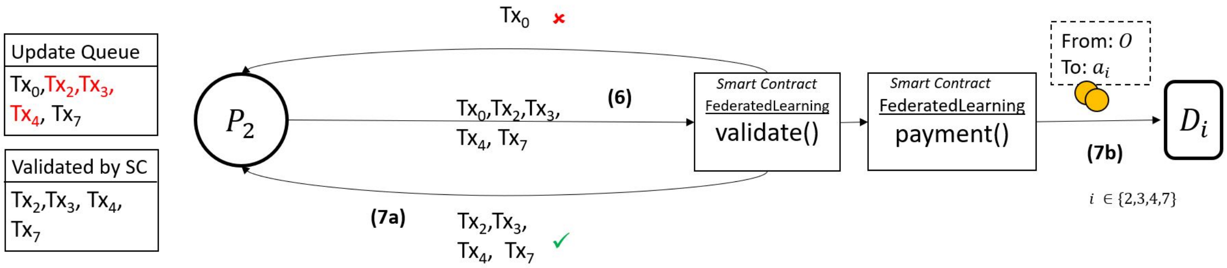


Fig. 4.3. The continued workflow of validating the δ via Smart Contracts and paying successful candidates, as explained is Section 4.3.3 @2019 IEEE.

Tableau 4.1. Parameters required in a transaction upload from device D_i @2019 IEEE.

Parameter	Purpose	Size	Section
TxID	Unique identifier of the transaction	256 bits	
a_i	Address of device D_i	160 bits	
$H(\delta_i)$	SHA256 Hash of binary representation of training model weight updates	~256 bits	Section 4.3.3
n_i	The data cost – number of data points used to calculate the model update	16 bits	Section 4.3.3
v_i	The current version k of T_k	16 bits	Section 4.3.4
v, r, s	Signature of hash of the transaction	65 bits	

4.3.1. System And Blockchain Architecture

For our system design, we are using EOS Blockchain, a public blockchain with no transaction fees which further incentivizes its use by users (44) (5). EOS uses a set of 21 *producers* to create blocks simultaneously, creating an extremely scalable blockchain able to process millions of transactions per second. In our system, the model owner $\mathcal{O}$ has full liability of payment for the device and producer work, as opposed to devices D needing to pay for their transactions (60), or miners to reward devices out-of-pocket (46).

Our base implementation of Federated Learning is built with smartphone devices in mind who act as the users performing the training and sending in δ values. We have a model owner $\mathcal{O}$ who defines the initial model and distributes the reward.

A transaction in our system carries the information needed for the Federated Learning process. For a user D_i and a global model T_k, we define these transaction parameters below.

The *gradient upload* δ_i is the binary representation of the weight updates to the model. For a received model T_k, T_k^i is the result of training the model on local data $\boldsymbol{d_i}$ and our gradient is the difference $\delta_i = T_k^i - T_k$. Regardless of what data we're using, the value of δ_i will be quite large. For example, the single channel MNIST image data (30) is 1 MB per update δ_i; this value could decrease for non-image data, or increase for larger image data. The way we record these values within the blockchain is to store signed transactions in a table off-chain within the IPFS file system (11), and record only the hash of the gradient value $H(\delta_i)$ on-chain. This process is shown in Fig. 4.4. This means that when the Smart Contract validates the format of δ, it must use an oracle to access the value in the table off-chain. When the owner updates the model, it must grab the gradients from this same off-chain table by querying IPFS for the document with the same on-chain gradient hash.

The *data cost* $|\boldsymbol{d_i}| = n_i$ of D_i is the number of datapoints used for training the model T_k to obtain δ_i. The amount rewarded to D_i for its gradient δ_i is proportional to the data cost n_i.

The *version* of the model being used is the integer value k of the current Global Model T_k. If the uploaded version v_i does not match the current version k, either the update value δ_i needs to be adjusted, or the value δ_i should be dropped.

The *address* a_i is the network address of the device D_i where payment is to be sent.

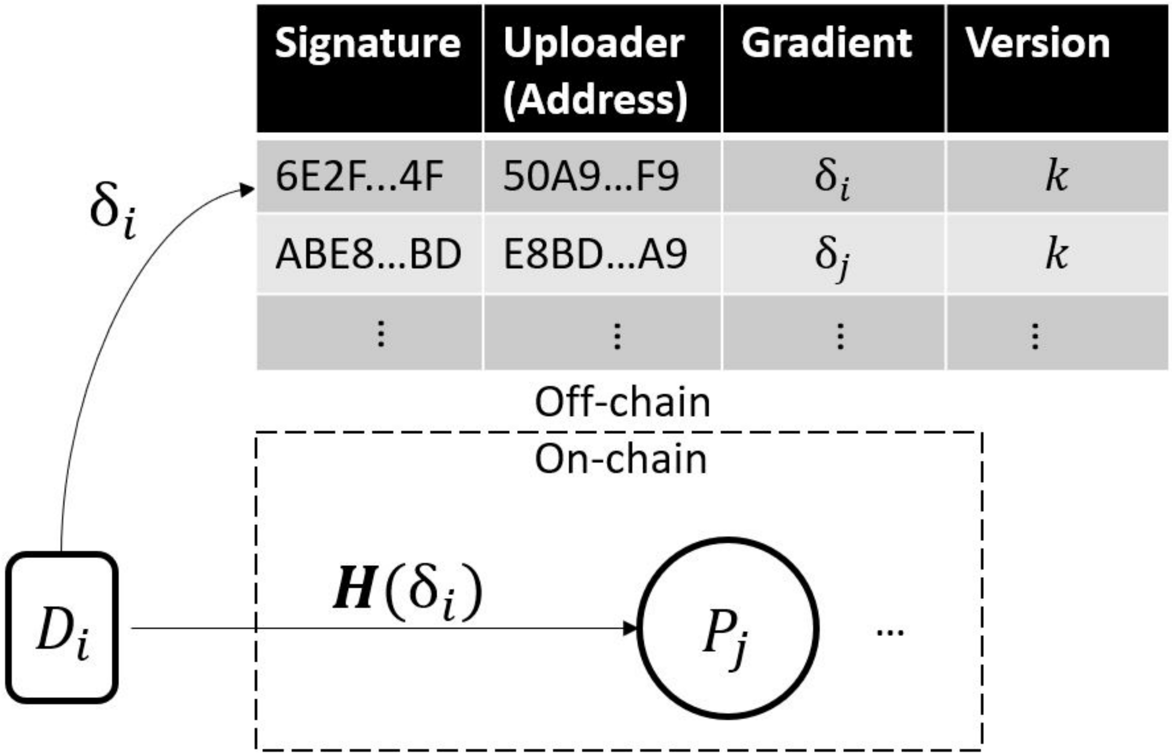
In addition to these values, each transaction has a *transaction id* to uniquely identify a transaction, and a *signature* created by the user's private and public key pair and denoted by v, r, s values. A summary of the transaction values is available in Table 4.1.

4.3.2. Smart Contracts

We propose the creation and usage of three Smart Contracts in our system.

`UploadGradient` – This Smart Contract verifies that every parameter as described in Table 4.1 is present in the transaction. It compares the hash of the transaction update value δ_i with the same value on the off-chain IPFS record as per Fig. 4.4, ensures that the hashes are equal, that the true value of δ_i follows the required size and format requirements for T_k as set by $\mathcal{O}$, and verifies the submitted version v_i is the same value as the last version value v_k sent out by $\mathcal{O}$ (Section 4.3.1). Once a transaction is validated as being a proper update transaction, the Smart Contract returns `true`.

`Payment` – Once a transaction from device $D_i \in \mathcal{D}$ is validated, this Smart Contract is triggered to send payment proportional to the data cost n_i to address a_i. Since we are using



Signature	Uploader (Address)	Gradient	Version
6E2F...4F	50A9...F9	δ_i	k
ABE8...BD	E8BD...A9	δ_j	k
⋮	⋮	⋮	⋮

Fig. 4.4. A device uploads the gradient value to an off-chain table within the IPFS file system where it is later accessed by the Smart Contract for validation, and the owner for gradient aggregation. Only the hash of the gradient remains on-chain to any Producer @2019 IEEE.

EOS, this payment would take the form of a EOS tokens of which $\mathcal{O}$ has given prior approval to the Smart Contract to transfer to devices.

`FederatedLearning` – This is the only Smart Contract a user may call; it verifies a user's role per a set of restrictions defined in Section 5.2.3 and forwards its transaction to the `UploadGradient` Smart Contract if the validation succeeds. The parameters required by this function for a device D_i are $H(\delta_i), n_i, v_i$ and a_i, and is forwarded to later calls of `UploadGradient()` and `Payment()`.

4.3.3. System Design And Workflow

The kth model is calculated from applying all of the model updates currently in the blockchain up to the kth block. We define $\mathcal{D}$ to be the set of all devices and $\mathcal{P}$ to be the set of all producers. As in Fig. 5.3, step (1), each device $D_i \in \mathcal{D}$ has a copy of T_k given to it by $\mathcal{O}$, along with the current version v_k; this is defined as *round k*. We walk-through the process

of training the next model, validating the transactions and paying the users, referring to Fig. 5.3 and 5.4.
For each device $D_i \in \mathcal{D}$ upon receiving the kth model T_k, (2) D_i trains the model T_k off-chain using it's local data $\boldsymbol{d_i}$ of size n_i to get an updated model T_k^i. The gradient is then calculated as $\delta_i = T_k^i - T_k$. (3) The values of δ_i, the size of $\boldsymbol{d_i} = n_i$, the device address a_i and the version v_k are set as parameters to the Smart Contract `FederatedLearning()` together with a TxID and a digital signature. (4) The device D_i sends this transaction on-chain to any producer P_j to which D_i is connected. (5) Each producer $P_j \in \mathcal{P}$ adds the transactions received by the devices to their transaction queue (i.e., pending transactions) to be verified for Block k. (6) Each producer executes each transaction in its queue, which involves a call to `FederatedLearning()` to validate the user role, then a call to `UploadGradient()` where details about each transaction are validated, such as the correct format for δ_i, the correct version v_i, and that the address a_i exists. (7a) Once validated, this transaction is added to the next block; (7b) the device $D_i \in \mathcal{D}$ who submitted a valid transaction is rewarded through a call to `Payment()` an amount proportional to the data cost n_i via the submitted address a_i.

4.3.4. Global Model

The *initial model* T_0 has all weights initialized to normally distributed values with mean 0 and variance 1; therefore, we define each weight $w \sim \mathcal{N}(0,1)$.
To calculate T_1, the owner $\mathcal{O}$ applies the aggregate of all δ_i updates from the blockchain where *version* $= 1$; we denote the number of such updates by b_1. We define w_l as being the lth weight in T_0, and $\delta_{i,l}$ as the lth weight of the ith gradient value; the training model will apply the update

$$w_l \leftarrow w_l + \eta \cdot \bar{\delta}_l = w_l + \eta \cdot \frac{1}{b_1} \sum_{i=1,\ldots,b_1} \delta_{i,l}$$

to each $w_l \in T_0$, where η is the *learning rate* (119).
Similarly, to calculate T_{k+1}, the owner $\mathcal{O}$ applies the update

$$w_l \leftarrow w_l + \eta \cdot \bar{\delta}_l = w_l + \eta \cdot \frac{1}{b_{k+1}} \sum_{i=1,\ldots,b_{k+1}} \delta_{i,l}$$

to each $w_l \in T_k$. If the most recent version for which the uploads have been aggregated is version K, then T_K is known as the *Global Model.*

4.3.5. Data Validity And Quality

The validation check we have defined earlier requires producers to trust that the data cost value a device claims to have used is correct. We propose a new concept of tailoring a validation set to a device's data breakdown which we will call a *Class-Sampled Validation Error Scheme* (CSVES).

This proposed method, shown in Fig. 4.5, begins prior to training. We define the set of all classes available as $C = \{C_1, C_2, ..., C_p\}$ for p classes. For a device $D \in \mathcal{D}$, we define the set C_D as the set of all classes for which D has data. Then, $C_D \subseteq C$ because there may exist a class $C_i \in C$ such that for all local datapoints $d \in \boldsymbol{d}$, $d \notin C_i$. D sends the set C_D to $\mathcal{O}$ and receives a validation set with datapoints chosen only from the classes in C_D. Once received, D begins to train model T_k with the local training dataset $\boldsymbol{d}$ and the received validation set. During training, the model will intermittently apply the validation set to the training model and record the validation error; if the model is improving, we expect the validation error to decrease. At the end of training, D will send the validation errors alongside other parameters for the function call `UploadGradient()`. We modify this function to look at the general trend of the validation errors over training time; if the validation errors are decreasing, we say δ is a valuable and valid gradient update, and we reward D based on its data cost n.

This algorithm does not require any trust of the devices who can inflate their data cost n for a higher reward; however, it is possible for either a faulty gradient δ to appear valuable and thus be rewarded, or for a valid gradient δ to have an increasing validation error if the datapoints used by D don't reflect the data in the received validation set and thus not be rewarded.

The most obvious issue with CSVES is that it is only defined here for classification problems; it would be useful to find other similar schemes for tailoring validation sets based on non-classification data. This scheme has also not been implemented and tested, so it is still to be seen whether it is possible to filter out "garbage" data that does not pertain to the model at all, or conversely how accurate this scheme is at identifying valuable gradients.

4.4. Proof Of Concept

We made a small implementation of our design in order to test out the general workflow in practice. For this implementation, we used 15 training rounds of 10 local `Device` participants who performed the model training off-chain in Python, and sent transactions to the Hyperledger Fabric blockchain. Future plans include a larger implementation of this system with EOS blockchain.

The Proof of Concept seeks to answer whether a blockchain could work with Federated Learning implementation in Python to record and reward gradient uploads; since we're using a REST API to interact with Hyperledger Fabric, then any programming language that supports API calls can interact with our blockchain system.

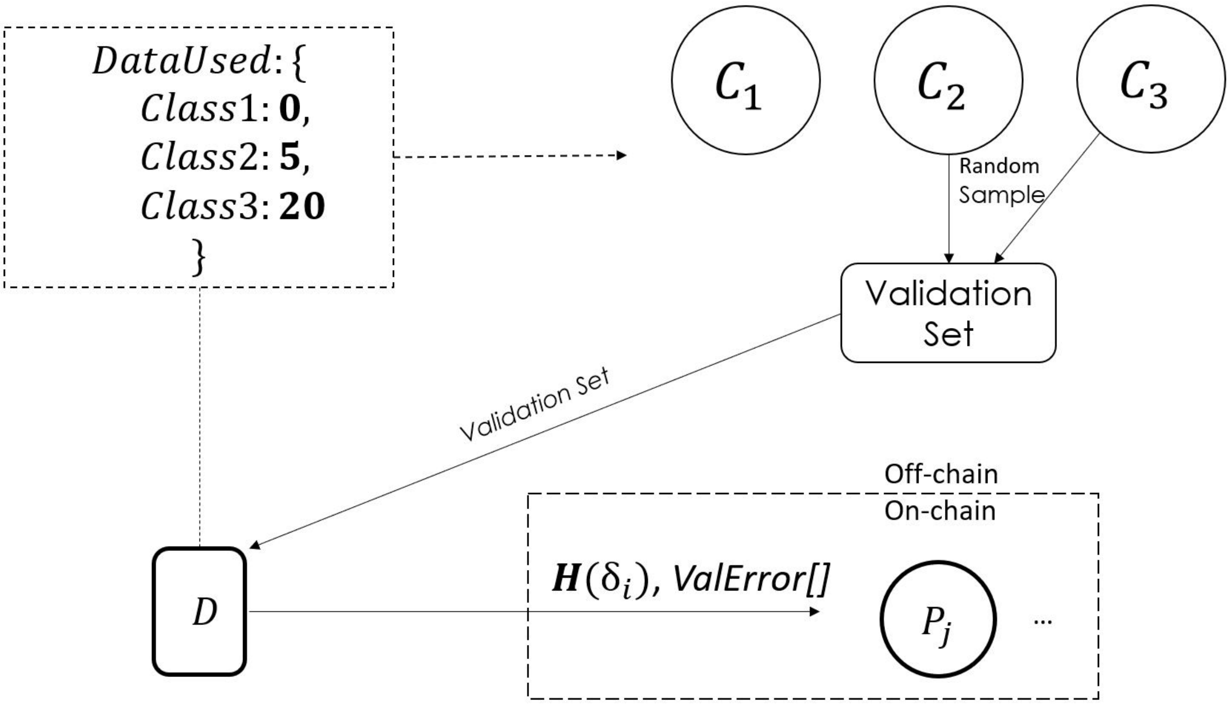


Fig. 4.5. Prior to training, a device $D \in \mathcal{D}$ sends a list of the classes for which it has data, and receives a validation set containing data from only those classes. Once the validation set is received, training proceeds and the set of validation errors throughout training is sent along with the gradient δ @2019 IEEE.

4.4.1. Hyperledger Fabric - REST API

For this implementation on Hyperledger Fabric we made use of Hyperledger Fabric Composer where we defined the files `model.cto` – where we define the *participants*, *assets* and *transactions*, `logic.js` – where we define the Smart Contract chain code, and `permissions.acl` – where we define the ruleset restricting/allowing the actions of the participants.

The `model.cto` defines the participants, assets, and functions of our system. We can have multiple `Device` participants, which make up the users in our Federated Learning process. Although training of the model occurs off-chain, we have a single instance of a `TrainingModel` asset which we use to keep track for which *version* we are currently uploading gradient values. A `Gradient` asset is linked to a `Device` participant, and has the *hash* of the gradient as in Fig. 4.4, the current training *version*, and the *dataCost* claimed by the *Device* participant. The `Token` asset is also linked to a `Device` participant, has the current training *version*, and a *value* which is equal to the claimed *dataCost*. The `UploadGradient()` transaction is the parameter description of our Smart Contract, requires a `Device` participant instance and the `TrainingModel` asset instance, and has the *hash*, *dataCost*, and *version* parameters which we've seen in the other assets.

The `logic.js` file is the *chaincode* of our application which is HyperLedger Fabric's version of a Smart Contract written in pure JavaScript. In this function, we have decided to combine the `Payment()` functionality within the same `UploadGradient()` function – the script validates the submitted parameters, creates the necessary assets, and rewards the user for their upload. The script follows these steps:

```
UploadGradient(tx.{Device, TrainingModel, version, hash, dataCost})
    (1) Validation
        (a) Verify that the uploaded tx.version is equal to the current
            tx.TrainingModel.version attribute.
    (2) Create new Gradient
        (a) Create a blank asset of type Gradient with unique id <tx.Device.deviceId_tx.version>.
        (b) Gradient.device ← tx.Device
        (c) Gradient.version ← tx.version
        (d) Gradient.hash ← tx.hash
    (3) Pay user via a Token
        (a) Create a blank asset of type Token with unique id tx.Device.deviceId_tx.version.
        (b) Token.value ← tx.dataCost
        (c) Token.Device ← tx.Device
```

Within Hyperledger Fabric, the *deviceId* of a `Device` is used in the same way as the *address* a_i from our design. We leave it for future work to both implement and test the Class-Sample Validation Error Scheme and to use an Oracle to check the off-chain format of the gradient δ, both as part of the *Validation* phase.

Our `permission.acl` file defines the *allowance* of participants of which actions they can perform. We set the following rules:

- Devices have no access to create or modify `Gradient` assets unless submitting a transaction to `UploadGradient()`
- Devices have no access to create or modify `Token` assets unless submitting a transaction to `UploadGradient()`
- All participants have READ access to the all `Gradient` assets.

This ruleset ensures that participants only see the information they need to see, which are at most all the gradients being uploaded; they do not need to see the rewarded tokens.

To run and interact with the blockchain locally, we deployed the blockchain with a local REST API. Then, we could perform off-chain training and data storage with Python while sending calls to the API to read the blockchain data, getting the list of active participants for whom we need to train, and posting transactions with calls to `UploadGradient()`.

4.4.2. Implementation Worflow

We have Python and Hyperledger Fabric working together to achieve Federated Learning using the following workflow.

(1) (a) Create the Initial Model in Python as per Section 4.3.4.
(b) Create the `TrainingModel` asset in Hyperledger Fabric with *version*: 0.

(2) (a) Create D local devices in Python, each with a unique *id*.
(b) Create D `Device` participants with the same *id* values as in Python.

(3) For version $v = k, k \in \{0,...,V\}$, perform the Federated Learning Process:

```
for $D \in \mathcal{D}$ do
    $T'_k \leftarrow \texttt{train}(T_k)$
    $\delta \leftarrow T'_k - T_k$
    · Write $\delta$ off-chain with version and dataCost
    · Upload $\delta$, version and dataCost to Hyperledger Fabric, creating a Gradient asset and a Token asset
end for
```

(4) Average all gradients with *version* $v = k$ to obtain $\bar{\delta}_k$.

(5) Apply the federated aggregation on T_k to obtain

$$T_{k+1} \leftarrow T_k + \eta \cdot \bar{\delta}_k$$

where $\eta \in (0,1]$ is the *aggregation factor*.

(6) Increment `TrainingModel`.*version* $v \leftarrow k + 1$ on Hyperledger Fabric.

4.4.3. Results

For initial evaluation, our setting consist of $n = 10$ clients each with an uneven and overlapping split within the range of 0.5 to 0.1 of the MNIST dataset, and running 15 rounds, we obtain the accuracy metrics of the Global Model shown in Fig. 4.6. We can see from Fig. 4.6 and Fig. 4.7 that the model improves but quickly plateau with small dips in accuracy without deviating significantly from a centralized approach. This confirms that the blockchain does not interfere with the Federated Learning process, while recording and rewarding devices for their data in Hyperledger Fabric which we can easily view from the browser's REST API dashboard. This plateau could be due to the lack of diversity of the devices per round, or the lack of diversity in the datasets of each device per round. Ideally, the Federated Learning training model would see new data every round; this type of experimentation is left for future work. Another reason may be that the number of steps taken by the training model is not optimal; we attempted to make every device train the same way, fixing the number of training iterations to do so. The small dips in accuracy could be due to over-fitting on the part of each device as the model become more accurate. For these two reasons, more optimal methods of enforcing a standard training process to avoid over-fitting and to reach training optimization are left for future work (32).

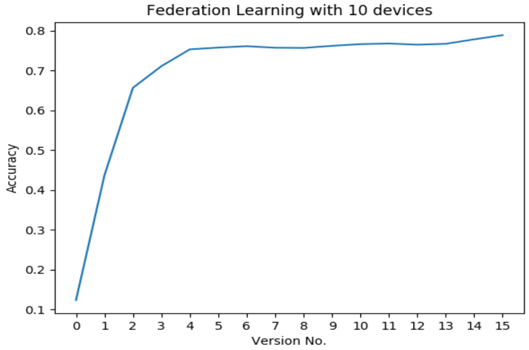


Fig. 4.6. Graphical results of training accuracy of 10 devices over 15 rounds @2019 IEEE.

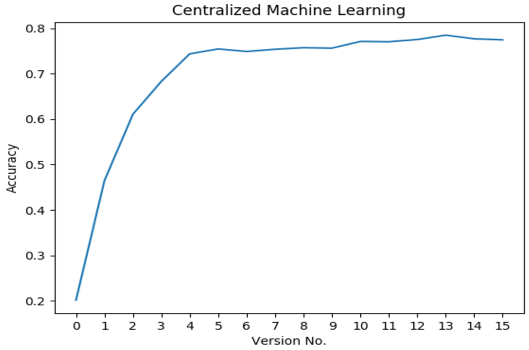


Fig. 4.7. Graphical results of training accuracy of centralized dataset over 15 rounds @2019 IEEE.

4.5. Scalabilty

Transaction latency is an essential characteristic for a blockchain platform. The number of transactions executed per second is a significant concern for most blockchain platforms because of the blockchain transactions requiring each node in the network to come to a consensus for anything to pass through. (125)(132). The core underlying issues rise from limitations of low throughput, high transactional latency, and increasingly high resource needs when used in a federated setting (132). The storage space requirements for the Blockchain will continue to increase as the number of clients increase. Such large storage requirements may eventually result in few prominent clients to take control of majority of the nodes and may lead to cheating whereas the other nodes are not able to detect this fraud (18).

By using EOS blockchain, we hope to address this issue since EOS is designed to support high scalability. EOS blockchain used in our implementation claims the ability to support millions of transactions/clients per second owing to their distributed proof of stake (DPOS) mechanism. With EOS, it takes just 0.5 seconds to create new blocks in the network which is comparatively very less than the existing blockchain platforms. Federated blockchain systems can expect participation of numerous clients and hence EOS is a feasible solution in terms of scalability.

4.6. Future Work

Moving forward, further implementation, testing and analysis of the proposed validation model CSVES is left as future work to evaluate whether CVES is an accurate scheme for determining the quality or usefulness of local data used for training the model. We also plan to investigate other validation schemes that can accurately determine how much payment a gradient upload should be rewarded, either based on verified number of data points or on evaluated data model improvement. Finally, we have acknowledged the value of having a standardized form of training such that two devices with the same data calculate the same gradients; such a standard will lead to consistency in uploaded results and fairness in rewards.

4.7. Conclusion

In this proposal, we addressed the problems of data privacy, security, and fair reward in distributed machine learning using blockchain and Federated Learning. An in-depth workflow was presented for scalable recording and rewarding of gradients using a combination of blockchain and off-chain databases of records. We have also proposed CSVES to validate and verify gradients to determine a reasonable device reward. We implemented a Proof of Concept with a small set of clients and rounds to demonstrate that the blockchain does not interfere with the federated learning aggregation, while limiting the number of uploads and validating the claimed data cost per device. Finally, we composed a list of aspects of Federated Learning and Blockchain that require more in-depth study for implementation as part of future work. With the proposed system, individuals benefit by retaining ownership and receiving incentives for their data, and model owners benefit from access to a larger and more diverse set of client data, leading to more robust and higher performing Machine Learning models.

Chapitre 5

CausalFedBlock : Blockchain For Federated Causal Invariance Learning With Fair Incentivization

Federated learning has become increasingly popular as it facilitates collaborative training among multiple clients under the coordination of a central server, while preserving the client data privacy. In practice, some of the main challenges posed to federated learning is model robustness, out of distribution generalization, fair incentivization and vulnerability to various privacy attacks. We propose a novel blockchain based causal approach to developing a robust federated ecosystem that achieves generalization beyond the limited set of client environments accessible to the server during a specific round of training, with the capability to extrapolate to new and unseen environments in server/client side enhancing fair incentivization for all participants. For real world deployment of federated systems, participant incentivization based on causal/invariant learning as opposed to associational learning methods will prove to be extremely beneficial in terms of fairness, privacy and robustness. In this work, fairness refers to equitable incentivization.

5.1. Background

5.1.1. Domain Generalization

Robustness to unseen domains is of utmost importance in federated setting and hence highly dependent on the fields of domain adaptation, domain generalization and invariant learning, all of which aim at enhancing model performance on a test distribution distinct from that of the training distribution. In the field of domain adaptation, the learner exploits the access to labeled data from the training domain and unlabeled data from test domain and performs well on the test domain. Domain generalization methods (77)(7)(53) in supervised learning use labeled data from multiple training domains while not requiring any unlabeled test data and learn models that generalize well to unseen domains. Domain generalization based methods seem to outperform domain adaptation methods in several use cases but have not been explored well in federated setting, which is the objective of this work. In our work, we rely on a recent domain generalization framework called invariant risk minimization (IRM) (77). The IRM uses the following principle to perform well on unseen domains: rely on features whose predictive power is invariant across domains, and ignore features whose predictive power varies across domains.

5.1.2. Blockchain Based Incentivization

The original concept of Federated Learning sought to allow users to keep ownership and privacy of their data during the model training process by only returning the δ update based on local data and not the local data itself(57)(83). This approach can however lead to privacy breaches by analyzing the δ output of users (103) (83)(38); a proposed solution to these privacy issues is to add noise to the updates at a negligible loss(83)(103)(38). In the absence of blockchain security, distributed learning techniques have been suggested to use homomorphic encryption to protect training data (51)(101); however, Federated Learning's approach to only uploading δ updates ensures privacy, and the blockchain ensures security(57) (83) One implementation of Federated Learning known as *BlockFL* uses blockchain to reward users for their local updates proportional to how many local data points are used(56). The blockchain is meant to enhance privacy and security of local the update δ for the user, and validity of δ to the model(56). In (56), the reward system may not produce a lasting solution since many miners would be paying users "out-of-pocket". These rewards are proportional to the number of data points used for the update, and may be inflated by a malicious node; it is proposed for a miner to analyze computation time of users to combat this, though even that could be misrepresented(56). Though the BlockFL implementation warns that two miners simultaneously creating a block may cause two separate blockchains, using the "Longest Chain" rule will mitigate this (89). *DeepChain* looks into using blockchain for Distributed Deep Learning applications(51) as a means to keep both the data and the model private and secure. DeepChain proposes an incentive-based blockchain mechanism where both *parties* who upload data to the model, and *workers* who process the data and update the model, get paid an amount π_P and π_W based on their *contribution* ω_P and ω_W for parties and worker respectively.(78)proposed solutions to some of the issues faced in the previous approaches to improve privacy, access control and storage.

5.1.3. Invariant Risk Minimization

Consider datasets D_e, consisting of features of the feature vector ($x \in \mathcal{X}$) and the label ($y \in \mathcal{Y}$), from multiple training domains $e \in \mathcal{E}_{tr}$. (77) proposes to use these multiple datasets to learn a predictor $f : \mathcal{X} \rightarrow \mathcal{Y}$ that minimizes the maximum risk over all the domains $\mathcal{E}$ which implies $\min_f \max_{e \in \mathcal{E}} R_e(f)$ where $R_e(f) = \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{P}_e}[l(f(\mathbf{x}), y)]$ is the risk under domain e for a convex and differentiable loss function l. The equations is given by

$$\min_{\Phi:\mathcal{X}\rightarrow\mathcal{Y}} \sum_{e\in\mathcal{E}_{tr}} R_e(\Phi) + \lambda \left\| \nabla_{w|w=1.0} R_e(w \cdot \Phi) \right\|^2$$

where $\lambda \in [0, \infty)$ is a regularizer balancing between the first term(standard ERM), and the invariance of the predictor $1 \cdot \Phi(x)$. $\Phi : \mathcal{X} \rightarrow \mathcal{Y}$ is an invariant predictor, $w = 1.0$ is just a dummy classifier, the gradient norm penalty measures the optimality of the dummy classifier at each domain e.

5.1.4. Fairness In Causal Invariance Learning

In light of recent interest in ML community on how robust machine learning methods relate to algorithmic fairness, (6) studied whether algorithms from robust ML can be used to improve the fairness of classifiers that are trained on biased data and tested on unbiased data.In their experiments, they show that IRM(77) achieves better out-of-distribution accuracy and fairness than Empirical Risk Minimization (ERM) methods. (27) proposed another domain-invariant learning framework that incorporates Environment Inference to directly infer partitions that are maximally informative for downstream Invariant Learning with established connection to algorithmic fairness, which enables accuracy improvement and calibration in a fair prediction problem.

5.1.5. Federated Causal Invariance Learning

With the above stated benefits of IRM in the field of learning, it is very evident that applying it to a federated setting will enhance robustness, privacy and fairness of the final learned model. (37) proposed an approach for the same in a federated setting and demonstrated the improvement in out of distribution accuracy as well as privacy of the final learned model.

5.1.6. Data Sharing Strategy With Blockchain

The rise of Big Data and Internet of Things (IoT) technology has led to several distributed data stored in a central repository which impose huge processing and computing requirements. Due to higher memory, disk capacity, processing capability, I/O throughput, and storage, distributed approaches are preferred over centralized approaches for Big Data, and are particularly good for parallel processing (120)(42). Also, distributed learning algorithms have their foundations in ensemble learning which helps build a set of classifiers to improve the accuracy of a single classifier. An ensemble approach merges with that of a distributed environment since a classifier is trained onsite, with a subset of data stored in it.(63) Hence, the older practice which was so far concentrated on monolithic data sets from where machine learning algorithms generate a single model is now getting phased out with distributed machine learning algorithms (120)(42)(63)

Though Blockchain appears to be a good solution for Big Data storage, the management and architectural implications to handle big data is not as well researched. (54).

5.2. Proposed Architecture

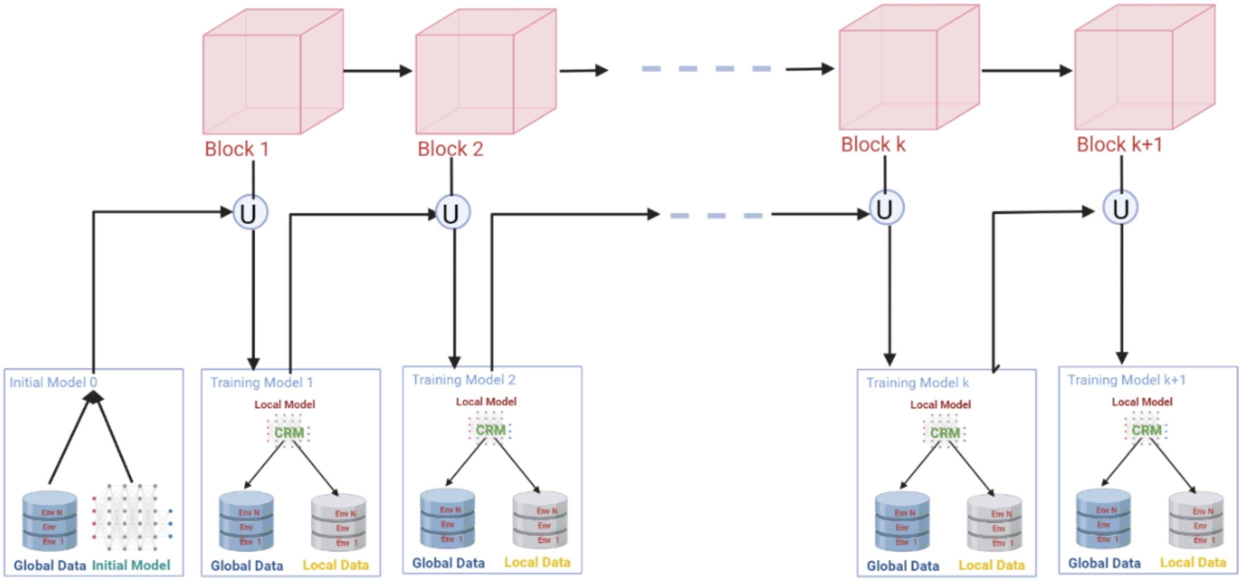


Fig. 5.1. Model T_{k+1} is calculated from applying the gradient values δ_i in Block $k+1$ to previous model T_k.

CausalFedBlock Architecture

In our proposed architecture, we are using Permissioned Blockchain which gives the owner $\mathcal{O}$ more control over who participates in the training process. This also gives the owner $\mathcal{O}$ full liability of payment for the device and miner work, as opposed to devices D needing to pay for their transactions (59), or miners to reward devices out-of-pocket (56). Although there may be a large number of devices uploading their δ values, the number of miners is restricted to however many the owner $\mathcal{O}$ designates the network needs; in practice the number of miners should be far less than the number of devices since the transactions are infrequent, only arriving once per device per training round at most. In our implementation, we have a model owner $\mathcal{O}$ who defines the initial model, shares the global data and distributes the reward. The *initial model* T_0 has all weights initialized to normally distributed values with mean 0 and variance 1; therefore, we define each weight $w \sim \mathcal{N}(0,1)$.This initial model, defined as model-0, is seen as a black box to all users except O; we do not allow for any device or miner to view the model.

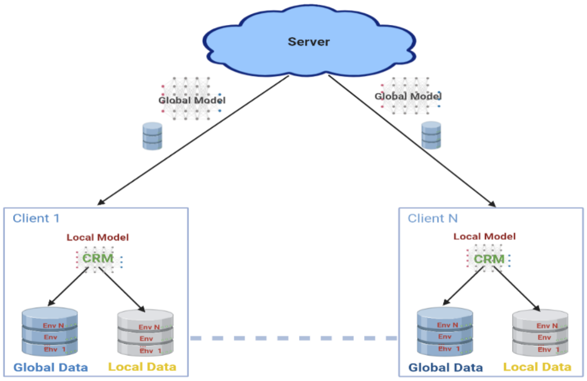


Fig. 5.2. Causal Federated Learning with Global Data Sharing Strategy
An Overview of Data Sharing Strategy

(37) proposed an approach to apply invaraint risk minimization on a federated setting based on which we built our pipeline for causalfedblock. To enhance learning of invariant features on each round of training, the model owner shares a small subset of global data containing a distribution over all the classes/enviroments from the server to block 1 during the initialization stage along with model-0.The local model of each client is learned by minimizing the empirical average loss as well as regularizing the model by the gradient norm of the loss for both the shared data from the owner(global environment) and private data from each client(Local Environment). This enhances the learning of invariant features common to both the client and global data environments without losing the privacy of client side data. In order to both keep the model from unaltered visibility from devices while allowing devices to train on the model, we employ homomorphic encryption to the model via the Paillier Cryptosystem which is commonly used in distributed deep learning(128). This symmetric cryptographic scheme has the property such that $E(x + y) = E(x) \cdot E(y)$, where $E(x)$ is the encryption of x. Since for a given weight in our training model we ultimately want to set $w \leftarrow w + \eta \cdot \bar{\delta}$(120), we can achieve this by setting $E(w) \leftarrow E(w) \cdot E(\eta \cdot \bar{\delta})$. To calculate model-1, a miner applies the aggregate of all δ_i updates from the 1st block; we denote the number of updates in block 1 by b_1. We define w_l as being the lth weight in T_0, and $\delta_{i,l}$ as the lth weight of the ith gradient value; the training model will apply the update

$$w_l \leftarrow w_l + \eta \cdot \bar{\delta}_l = w_l + \eta \cdot \frac{1}{b_1} \sum_{i=1,\ldots,b_1} \delta_{i,l}$$

to each $w_l \in T_0$, where η is the *learning rate*(120).

Similarly, to calculate model-$(k + 1)$, the owner $\mathcal{O}$ applies the update

$$w_l \leftarrow w_l + \eta \cdot \bar{\delta}_l = w_l + \eta \cdot \frac{1}{b_{k+1}} \sum_{i=1,\ldots,b_{k+1}} \delta_{i,l}$$

to each $w_l \in T_k$. If there are currently K blocks in the blockchain, then model-K is the *Global Model*; this process is shown in Figure5.1. The consensus protocol we elect to use is that of *Byzantine Fault Tolerance* (BFT)(117)(110). This protocol has a set of *endorsement peers* who each run a transaction, and each output whether a transaction is valid or not. The BFT endorsement protocol requires $3f < n$, where f is the number of faulty nodes and n is the total number of nodes taking part in BFT(110). In other words, we need more than 2/3 of the endorsement peers to agree on the validity of the transaction in order for consensus to be reached. Once consensus is determined for each transaction, the next Block k is created.

5.2.1. System Design

The system design is inspired from (78). The kth model is calculated from applying all of the model updates currently in the blockchain up to the kth block. We define $\mathcal{D}$ to be the set of all devices and $\mathcal{M}$ to be the set of all miners. As in Figure 5.3, step (1), each device $D_i \in \mathcal{D}$ has a copy of T_k given to it by O, along with the current version v_k; this is defined as *round* k. We walk-through the process of training the next model, validating the transactions and paying the users, referring to Figure 5.3 and 5.4.

For each device $D_i \in \mathcal{D}$ upon receiving the kth model T_k, (2) D_i trains the model T_k off-chain using it's local data $\boldsymbol{d_i}$ of size n_i to get an updated model T_k^i. The gradient is then calculated as $\delta_i = T_k^i - T_k$. (3) The values of δ_i, the size of $\boldsymbol{d_i} = n_i$, the device address a_i and the version v_k are set as parameters to the Smart Contract `UploadGradient()` together with a TxID, a timestamp and a digital signature. (4) The device D_i sends this transaction on-chain to its nearest miner M_j. (5a) Each miner $M_j \in \mathcal{M}$ adds the transactions received by the devices to their transaction queue, and simultaneously broadcasts the transactions to other miners; (5b) each miner receives the remaining transactions from other miners and add them to the final queue of transactions to be verified for Block k. (6) Each miner executes each transaction in its queue, which involves a call to `UploadGradient()` where details about each transaction are validated, such as the correct format for δ_i, the correct version v_i, and that the address a_i exists. (7a) Once validated, this transaction is added to the miner's pending queue for the next block; (7b) the device $D_i \in \mathcal{D}$ who submitted a valid transaction is rewarded an amount proportional to the data cost n_i via the submitted address a_i.

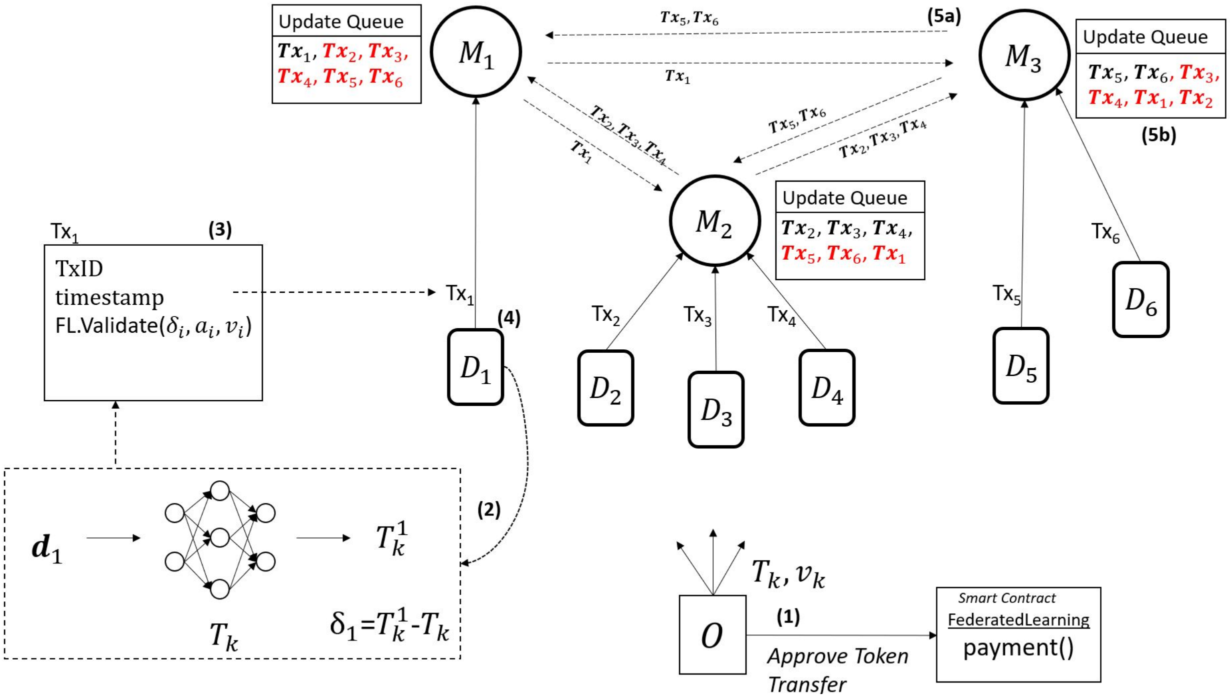


Fig. 5.3. The workflow of round k begins with (1) $\mathcal{O}$ distributing T_k and the version v_k to all devices, and simultaneously approving token transfer to the Payment() Smart Contract; (2) each device D_i uses a local dataset of size n_i and global dataset of size n_g to calculate gradient δ_i(57); (3) the values δ_i, n_i – the number of data points, a_i – the address of D_i, and v_i – the version of the model used, are packed into a Transaction, along with a TxID, timestamp and a Smart Contract function call to `UploadGradient()`; (4) each device D_i sends its transaction to its closest miner M_j; (5a) each miner M_j who received Tx_i validates the transaction and adds it to its queue, while simultaneously broadcasting the received transactions to all other miners and (5b) the miners who receive the remaining transactions have their final queue of transactions for training model T_k.

CausalFedBlock Workflow

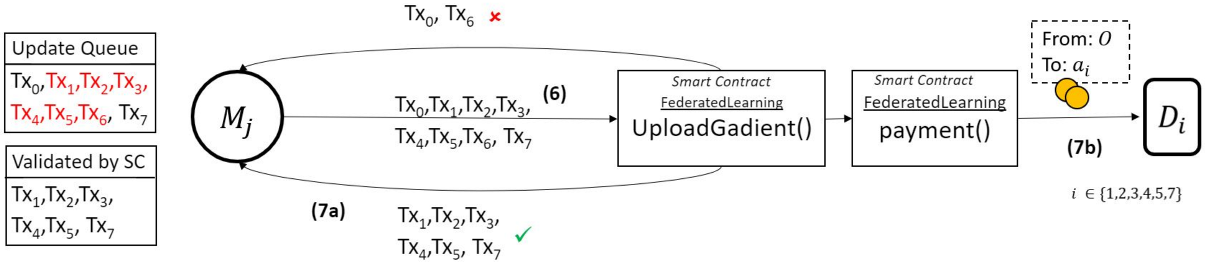


Fig. 5.4. The workflow continues with (6) miner M_j running each transaction in its queue, which involves a call to `UploadGradient()` where the format of δ_i, the correct version number v_i, and the existence of the address a_i are all checked; (7a) the transactions that run without errors are added to the miners queue for the next block and (7b) the devices who sent in the transaction are rewarded an amount of tokens proportional to n_i – the number of datapoints used for training.

CausalFedBlock Incentivization

5.2.2. Computation And Storage

A device $D_i \in \mathcal{D}$ must calculate it's update value δ_i locally since we assume we cannot use any miner $M_j \in \mathcal{M}$ to aid in computation since sending local data would break the data privacy benefits of Federated Learning. In order to avoid the need for Mobile Edge Computing (72), it is assumed that a device $D_i \in \mathcal{D}$ has enough processing power and memory to calculate an updated model T_k^i from model T_k; this may not be realistic if the size of T_k becomes much larger than what we had considered. It is not assumed that D_i can store the entire blockchain in-memory, only the block headers are useful to D_i along with the Merkle Path to the reward transaction of D_i.

5.2.3. Restrictions For Permissioned Blockchain

This blockchain system is partially private, since all block data needs to be visible to all those that partake in the blockchain, but to nobody outside the system. We define a *member* x of the blockchain as either a device $x \in \mathcal{D}$, a miner $x \in \mathcal{M}$ or the model owner $x = \mathcal{O}$. The only restriction we put is that $\mathcal{O} \cap \mathcal{D} = \varnothing$, meaning $\mathcal{O}$ doesn't take part in the Federated Learning training process and none of the devices take part in the model aggregation process. To achieve this, we are using a *permissioned blockchain* (21) for this task, and specifically we'll be working with Hyperledger Fabric (21) which has *permissions* which is a ruleset which defines which members have access to which actions. In this ruleset, we define the owner $\mathcal{O}$ as not being able to send any transactions that call the `UploadGradient()` Smart Contract, thus impeding it from contributing its own data to the process. Furthermore, we define a rule to limit a device's only action as sending in a transaction with a call to `UploadGradient()`.

At any time, the owner $\mathcal{O}$ can amend this ruleset in order to complete the training process by stopping all devices from uploading gradients. In addition to having a ruleset to restrict/allow actions from members of the network, a permissioned blockchain gives the owner $\mathcal{O}$ full control over who joins the network in the first place, and can revoke access to joined devices at any time.

5.2.4. Addressing Privacy Leakage Of Global Data Using Access Control Implementation

In the BlockFL architecture proposition (56), the blockchain stored only the aggregated updates of the local devices, and not the data itself. Even then, Access Control is an important part in Blockchain systems that are not fully public (74)(75)(123). In the context of CausaFed setting, our objective is to regulate access of the global shared data portion and model updates in the Blockchain to authorized participating clients, being those with permission to view and send updates regarding the distributed model (83)(57)(56). This permission and authorization rule set can be varied as per the specific requirements of the data shared or the problem trying to be solved with the setup.

When Access Control in implemented, permission as to who can view the data is set by the owner of the data; however, the permission is granted by an entity that this owner trusts (74) - in this case being the model originator (83)(57)(56). Two conclusions may arise. Since the model updates are all aggregated, viewing a single block would require permission to be granted to the entity by each and every device that has partaken in those model updates; in this case, it would be necessary for viewing permission to the appropriate parties be given in addition to the local model updates. Secondly, though a device may give permission for their global data portion to be viewed by appropriate parties, a malicious central device may deny access to these same parties; a blockchain approach to these permissions will ensure that permissions given by a device are the same permissions granted by the central device (74). Thus, in the presence of an Access Control system, it serves to hold both model updates (56) and global shared data access permissions (74)(75).

Following existing proposals for using Access Control in blockchain (74)(33), we use *Attribute Based Access Control* (ABAC) (49) in our approach which is a form of Access Control where permission is granted according to a set of rules; if an entity's attributes satisfy the conditions set by these rules, the entity is awarded access. This type of access would allow for all appropriate parties to be given access with a single ABAC policy stored in the blockchain transaction. Similarly, permission updates' revokation can be done via a new transaction (74). Whenever an entity attempts to access a block, an Access Control system verifies whether this entity's attributes satisfy the ABAC policy across all ABAC transactions, and grants the entity access to the block accordingly (74).

Another method of Access Control enforcement is through the use of smart contracts. Maesa, Mori & Ricci (74)(75) (123) (19) (21) have explored the use of smart contracts, such as those used in Ethereum (19), for enforcing Access Control. Similarly, Uchibeke, Kassani, Schneider & Deter (123) did substantial investigation on using Hyperledger smart contracts (21) for Access Control.

```
namespace org.ownername.federatedlearning

participant Device identified by deviceId{
 o String deviceId
}

asset Gradient identified by deviceVersionId{
  --> Device device
  o String deviceVersionId
  o String hash
  o Integer version
  o Integer dataCost
}

asset Global Shared Data portion identified by deviceVersionId{
  --> Device device
  o String deviceVersionId
  o String hash
  o Integer version
}

asset TrainingModel identified by modelId{
  o String modelId
  o Integer version
}

asset Token identified by deviceVersionId{
  o String deviceVersionId
  o Integer value
  o Integer version
  --> Device device
}

transaction UploadGradient{
  --> Device device
  --> TrainingModel trainingmodel
  o String hash
  o Integer dataCost
  o Integer version
}
```

Listing 5.1. `model.cto` – Participant, Asset and Transaction definition.

5.3. Results

5.3.1. Dataset

We use **Colored MNIST Rotated MNIST** and **Rotated FMNIST** as was used in the previous chapters for fair comparison of results.

5.3.2. Evaluation Setting

We use the same setting as was followed in previous chapters to ensure fair comparison of results. Client nodes and server nodes are run on separate computing nodes of compute canada cluster. All clients are bound to update the model in each epoch of training. Approximately 12 GB RAM was allocated per slurm job. Python 3.7 is the language used for all the code implementation. Pytorch 1.10 is the machine learning library employed for all the prototypes. The network architecture and learning rates are selected based on model performance during intial rounds of training.

Tableau 5.1. Network Architecture

Architecture	**No of Layers**	**Kernel spec**	**Learning Rate**
LeNet	5	(5x5), (2x2)	0.003
AlexNet	8	(11x11), (5x5), (3x3)	0.0001
ResNet18	18	(7x7), (3x3)	0.0004

Federated Averaging denoted as FedAvg is considered the benchmark in our comparison study. The results shown are observed on 150 global epochs on a set of 10 clients with batch size set to 512. To test the variation of accuracy over each global epoch, we computed CV which is the coefficient of variation to measure the dispersion. It is given by the ratio of standard deviation to mean. Based on our analysis, CV coefficient values range from 0.04 - 5.2 for train results and 0.5 - 5.8 on test after epoch 5.

5.3.3. Out of Distribution(OOD) Test results

Tableau 5.2. Comparison of methods in terms of testing accuracy (mean ± std deviation)

Dataset	Arch	Fed-Avg	CausalFed	CausalFedGSD	CausalFedBlock
Colored MNIST	ResNet18	11±.15	65.62±.6	55.62±2.8	54.9±4.2
Rotated MNIST	ResNet18	82.7±.4	90.2±.8	85.2±2.5	84.5±4.9
Rotated FMNIST	LeNet	69±.19	74.6±.22	71.9±3.6	71.3±5.1

We observe that addition of blockchain to our system does not severely affect the test performance. However, scalabilty of the proposed approach is a concern as was specified in the previous chapter and hence future improvements will be considered to enhance better scalability of the CausalFedBlock Framework.

Table below shows a sample scenario for simulation of valid data acceptance rates via CSVES-Th with $\tau = 0.1$. Here we have simulated the training of 1000 instances on Rotated MNIST dataset with 5 epochs each of 1000 training points and 100 validation points.

Number of Classes	Instances	Accepted	% Accepted
1	9	5	%55.555
2	41	41	%100.000
3	112	112	%100.000
4	211	211	%100.000
5	240	238	%99.167
6	203	203	%100.000
7	119	119	%100.000
8	50	50	%100.000
9	13	12	%92.308

Tableau 5.3. Leakage on inference attack testing accuracy (mean ± std deviation)

Dataset	Fed-Avg	CausalFed	CausalFedGSD	CausalFedBlock
Colored MNIST	79.45±.12	56.57±1.5	58.57±2.6	57.9±3.4
Rotated MNIST	85.24±.5	63.3±2.9	65.3±2.9	64.4±3.1
Rotated FMNIST	78.23±.9	53.55±3.5	54.55±4.9	53.7±5.2

The privacy leakage on each of the attacks is measured by testing the accuracy of attack model. The use of smart contracts to validate each gradient upload as well as the validation scheme employed helped in maintaining better privacy guarantees for the proposed system.

5.4. Conclusion And Future Work

In this work, we addressed the problems of data privacy, out of distribution generalization and fair incentivization in a federated setting using invariant learning and blockchain. An in-depth workflow with detailed architecture and system design was presented on the development of a federated model highly confident of predictions outside the client level train datasets with enhanced robustness to distributional shifts away from the training set giving weightage to learning a representation common to all participating clients relying upon features that affect the target in all participating client environments ignoring individual client dataset-specific correlations ensuring fair incentivization for paticipating clients. Also, our work is the first to blend the fields of blockchain, federated learning and causal/invariant learning, each of which makes incredible contributions to enhance the other.

For future work, improving model fairness will be of interest. In the Colored MNIST experiments the spurious covariate (colour) is easily inferable from image data using RGB channel information(6). Hence moving forward, we will focus on more biased datasets like the toxic comment detection task to evaluate the generalization and fairness properties of causalfedblock. One way would be to induce positive spurious correlation between the comment being about a particular demographic identity and comment toxicity construct across client domains followed by reversing that correlation strength in the owner/server domain.

Chapitre 6

Conclusion And Future Directions

In this work, we addressed the problems of data privacy, out of distribution generalization and fair incentivization in a federated setting using causal invariance learning and blockchain. With blockchain, we addressed the problems of data privacy, security, and fair reward in distributed machine learning using blockchain and Federated Learning. An in-depth workflow was presented for scalable recording and rewarding of gradients using a combination of blockchain and off-chain databases of records. We have also proposed CSVES to validate and verify gradients to determine a reasonable device reward. We implemented a Proof of Concept with a small set of clients and rounds to demonstrate that the blockchain does not interfere with the federated learning aggregation, while limiting the number of uploads and validating the claimed data cost per device. An in-depth workflow with detailed architecture and system design was presented on the development of a federated model highly confident of predictions outside the client level train datasets with enhanced robustness to distributional shifts away from the training set giving weightage to learning a representation common to all participating clients.

In the field of federated learning, the study of causal invariance learning and incentive mechanisms is still in its infancy, and there exist many open issues. We just point out some directions for the future studies.

6.0.1. Robustness

In our experiments, we observe that for most of the distribution shifts involving confounders or anti-causal variables, our approach with causal invariant learning gives close to the desired OOD solutions in the finite sample regime. While the results in using invariant learning techniques in federated setting seems promising, some problems of these methods have been studied in (105) on classification tasks. Eventhough the optimal solution succeeds in the linear case, (105) demonstrated simple conditions under which these approaches fail to recover the optimal invariant predictor. Specifically, Rosenfeld et al. (105) show that there exists a feasible solution which uses only the environmental features yet performs better than the optimal invariant predictor on all $e \in \mathcal{E}_{\text{all}}$ (105). Unless the test data are sufficiently similar to the training distribution, these methods fail catastrophically in nonlinear regime. (105). Hence moving forward, more studies need to be made on finding an optimal invaraint predictor in non linear regimes.

6.0.2. Privacy

We need to come up with distributed learning approaches that would ensure preserving the privacy of the shared causal representation of all participating clients. Under the federated learning framework, transfer knowledge is not only learned in a distributed manner, but also is typically not allowed to be exposed to any participant. Thus, we need to figure out precisely what each participant contributes to the shared representation in the federation and consider how to preserve the privacy of the shared representation. Downloading parts of global dataset to each client for model training in CausalFed-GSD setting can be considered as a violation to the requirement of privacy preserving learning in Federated setting for which we proposed access control techniques based on blockchain. Still, further studies need to be done to ensure privacy of some aspects of these approaches. Moving forward, we will focus on homomorphic encryption (41) and secret sharing (24) protocols to further enhance privacy.

6.0.3. Fairness in Incentivization

In the future, we will consider formulating a multi-dimensional metric instead of a single-dimensional metric which can appeal to multi-goals and multifunctionalities in one scheme. Moving forward, we will put more emphasis on cross-silo federated setting. Also, further implementation, testing and analysis of the proposed validation model CSVES on real world diverse datasets is left as future work to evaluate whether CVES is an accurate scheme for determining the quality or usefulness of local data used for training the model in real world settings. For an appropriate incentive design of high potential scenarios such as mobile edge computing, we will look into learning techniques such as graph neural networks, generative adversarial networks and multi-agent reinforcement learning.

6.0.4. Scaling

With the already specified advantage of EOS blockchain with scaling, we will further study possible approaches to enhance scalability of the proposed approaches. Some initial suggestions would be to rely on On-chain scaling and off-chain scaling techniques (132). Another favorable approach is edge computing which is suitable for high computational resources and storage requirements distributed at the network edge, offloading the Blockchain and mining computation from the power-limited nodes (132). We will study in detail existing approaches to various scalability solutions and come up with more viable scaling techniques consisting of on-chain, off-chain, sidechain, child-chain, and inter-chain-based solutions (55).

.

Références bibliographiques

[1] Kaggle: Your home for data science – competitions. `https://www.kaggle.com/competitions`. Accessed: 2019-03-25.

[2] The ocean protocol. `https://oceanprotocol.com/protocol/`. Accessed: 2019-03-25.

[3] Wibson: Don't give away your data for free. make a profit. `https://wibson.org/`. Accessed: 2019-03-25.

[4] Wibson: Don't give away your data for free. make a profit. `https://wibson.org/`. Accessed: 2019-03-25.

[5] Eosio. 2019.

[6] Robert Adragna, Elliot Creager, David Madras et Richard Zemel : Fairness and robustness in invariant learning: A case study in toxicity classification, 2020.

[7] Mathias Humbert Pascal Berrang-Mario Fritz Michael Backes Ahmed Salem, Yang Zhang : Mlleaks model and data independent membership inference attacks and defenses. 2018.

[8] J.R. Anderson : *Security Engineering: A Guide to Building Dependable Distributed Systems*. Wiley, Indianapolis, IN, USA, 2nd édition.

[9] Anonymous et Feb : 'new kid on the blockchain,'. *New Scientist*, 225(3009):7,. Available:.

[10] M. Arjovsky, L. Bottou, I. Gulrajani et D. Lopez-Paz : Invariant risk minimization. arXiv preprint arXiv:1907.02893,.

[11] J. Benet : Ipfs-content addressed, versioned, p2p file system. 2014.

[12] A.P. Bernstein et E. Newcomer : *Principles of Transaction Processing*. Morgan Kaufmann, Burlington, VT, USA, 2nd édition.

[13] Kreuter B Marcedone A-McMahan HB Patel S Ramage D Segal A Seth K Bonawitz K, Ivanov V : Practical secure aggregation for privacy-preserving machine learning. pages 1175–1191, 2017.

[14] Hubert Eichner Bonawitz Keith : Towards federated learning at scale: System design.

[15] K Zhang Brendan McMahan, Ramage Talwar : Learning differentially private language models without losing accuracy. 2017.

[16] C. Bucilua, R. Caruana et A. Niculescu-Mizil : Model compression. *In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, page 535–541.

[17] P. Buhlmann : Invariance, causality and robustness. arXiv preprint arXiv:1812.08233,.

[18] V. Buterin : A next generation smart contract and decentralized application platform. Accessed: May 13, 2019. Available:.

[19] Vitalik Buterin *et al.* : A next-generation smart contract and decentralized application platform. *white paper*, 2014.

[20] Melis C, Song E, De Cristofaro et V. Shmatikov : Exploiting unintended feature leakage in collaborative learning. *IEEE Symposium on Security and Privacy*, 2019.

[21] Christian Cachin : Architecture of the hyperledger blockchain fabric. *In Workshop on distributed cryptocurrencies and consensus ledgers*, volume 310, 2016.

[22] Jakub Konecny H.Brendan McMahan Caldas, Sebastian.

[23] H. Chang, V. Shejwalkar, R. Shokri et A. Houmansadr : Cronus: Robust and heterogeneous collaborative learning with black-box knowledge transfer. arXiv preprint arXiv:1912.11279 .

[24] W.T. Chang et R. Tandon : On the capacity of secure distributed matrix multiplication. *In 2018 IEEE Global Communications Conference (GLOBECOM), IEEE*, page 1–6.

[25] Minghao Zhao Zhenxiang Chen-Chong Zhi Gao Hongwei Li Yuan Tan Chuan Zhao, Shengnan Zhao : Secure multi party computation theory practice and applications. 2019.

[26] Ivan Beschastnikh Clement Fung, Chris JM Yoon : Mitigating sybils in federated learning poisoning. 2018.

[27] Elliot Creager, Jörn-Henrik Jacobsen et Richard Zemel : Environment inference for invariant learning, 2021.

[28] P. B uhlmann D. Rothenhausler, N. Meinshausen et J. Peters : “anchor regression: heterogeneous data meets causality. arXiv preprint arXiv:1801.06229,.

[29] Decentralized Machine Learning : Decentralized machine learning white paper, cited March 2019.

[30] L. Deng : The mnist database of handwritten digit images for machine learning research. 2012.

[31] C. Dwork : *Differential privacy: A survey of results, in: International conference on theory and applications of models of computation.* Springer.

[32] Y. Hua D. Estrin-V. Shmatikov E. Bagdasaryan, A. Veit : How to backdoor federated learning. 2018.

[33] Hamza Es-Samaali, Aissam Outchakoucht et Jean Philippe Leroy : A blockchain-based access control for big data. *International Journal of Computer Networks and*

Communications Security, 5(7):137, 2017.

[34] C. Esposito, A.De Santis, G. Tortora, H. Chang et K.-K.-R. Choo : 'blockchain: A panacea for healthcare cloud-based data security and privacy?'. *IEEE Cloud Comput*, 5(1):31–37,.

[35] M.A. Ferrag, M. Derdour, M. Mukherjee, A. Derhab, L. Maglaras et H. Janicke : 'blockchain technologies for the internet of things: Research issues and challenges,'. *IEEE Internet Things J*, 6(2):2188–2204,.

[36] Ari Juels Michael K Reiter Florian Tramèr, Fan Zhang et Thomas Ristenpart : Stealing machine learning models via prediction apis. 2016.

[37] Sreya Francis, Irene Tenison et Irina Rish : Towards causal federated learning for enhanced robustness and privacy, 2021.

[38] Ristenpart T Fredrikson M, Jha S : Model inversion attacks that exploit confidence information and basic countermeasures. pages pp. 1322–1333, 2015.

[39] Benjamin C. M. Fung, Ke Wang, Rui Chen et Philip S. Yu : Privacy-preserving data publishing: A survey of recent developments. *ACM Comput. Surv.*, 42(4), jun 2010.

[40] J.L. Gamella et C. Heinze-Deml : Active invariant causal prediction: Experiment selection through stability. *In NIPS*.

[41] C. Gentry : A fully homomorphic encryption scheme.

[42] Gennaro Costagliola Gianluca Roscigno, Giuseppe Cattaneo : The role of distributed computing in big data science: Case studies in forensics and bioinformatics. 2015.

[43] J. Gou, B. Yu, S.J. Maybank et D. Tao : Knowledge distillation: A survey. *International Journal of Computer Vision*, page 1–31.

[44] I. Grigg : Eos - an introduction. 2017.

[45] Daniel Ramage Seth Hampson Blaise Agüera y Arcas H. Brendan McMahan, Eider Moore : Communication-efficient learning of deep networks from decentralized data. 2017.

[46] M. Bennis H. Kim, J. Park et S.-L. Kim : On-device federated learning via blockchain and its latency analysis. 2017.

[47] C. Heinze-Deml, J. Peters et N. Meinshausen : Invariant causal prediction for nonlinear models. *Journal of Causal Inference*.

[48] G. Hinton, O. Vinyals et J. Dean : Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 .

[49] Junbeom Hur et Dong Kun Noh : Attribute-based access control with efficient revocation in data outsourcing systems. *IEEE Transactions on Parallel and Distributed Systems*, 22(7):1214–1221, 2011.

[50] F. X. Yu P. Richtárik A. T. Suresh J. Konečny, H. B. McMahan et D. Bacon : Federated learning: Strategies for improving communication efficiency. 2016.

[51] J. Zhang M. Li Y. Zhang J. Weng, J. Weng et W. Luo : Deepchain: Auditable and privacy-preserving deep learning with blockchain-based incentive. 2018.

[52] Anton Schwaighofer Joaquin Quionero-Candela, Masashi Sugiyama et Neil D Lawrence : Dataset shift in machine learning. 2019.

[53] B Schölkopf K Zhang, M Gong : Multi-source domain adaptation: A causal view. 2015.

[54] Elena Karafiloski et Anastas Mishev : Blockchain solutions for big data challenges: A literature review. *In IEEE EUROCON 2017-17th International Conference on Smart Technologies*, pages 763–768. IEEE, 2017.

[55] H. Kim et A. Mnih : Disentangling by factorising. *In International Conference on Machine Learning*, page 2649– 2658. PMLR.

[56] Hyesung Kim, Jihong Park, Mehdi Bennis et Seong-Lyun Kim : On-device federated learning via blockchain and its latency analysis. *arXiv preprint arXiv:1808.03949*, 2018.

[57] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh et Dave Bacon : Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[58] Nikola Konstantinov et Christoph Lampert : Robust learning from untrusted sources. 2019.

[59] A Besir Kurtulmus et Kenny Daniel : Trustless machine learning contracts; evaluating and exchanging machine learning models on the ethereum blockchain. *arXiv preprint arXiv:1802.10185*, 2018.

[60] B. Kurtulmus et K. Daniel : Trustless machine learning contracts;evaluating and exchanging machine learning models on the ethereum blockchain. 2018.

[61] L. Lamport : ‘password authentication with insecure communication,’. *Commun. ACM*, 24(11):770–772,.

[62] Latanyasweeney : k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10, 05 2012.

[63] Mu Li, David G Andersen, Alexander J Smola et Kai Yu : Communication efficient distributed machine learning with the parameter server. *In Advances in Neural Information Processing Systems*, pages 19–27, 2014.

[64] Ninghui Li, Tiancheng Li et Suresh Venkatasubramanian : t-closeness: Privacy beyond k-anonymity and l-diversity. *In 2007 IEEE 23rd International Conference on Data Engineering*, pages 106–115, 2007.

[65] X. Li, P. Jiang, T. Chen, X. Luo et Q. Wen : ‘a survey on the security of blockchain systems,’. *Future Gener. Comput. Syst*, 107:841–853,.

[66] Y. Li, Y. Yang, W. Zhou et T. Hospedales : Feature-critic networks for heterogeneous domain generalization. *In International Conference on Machine Learning*, page 3915–3924. PMLR.

[67] I.-C. Lin et T.-C. Liao : ‘a survey of blockchain security issues and challenges,’. *Int. J. Netw. Secur*, 19(5):653–659,.

[68] T. Lin, L. Kong, S.U. Stich et M. Jaggi : Ensemble distillation for robust model fusion in federated learning. *In 34th Conference on Neural Information Processing Systems*.

[69] Y. Liu, Y. Kang, C. Xing, T. Chen et Q. Yang : A secure federated transfer learning framework. *IEEE Intelligent Systems*, 35:70–82.

[70] Y. Lu : ‘blockchain: A survey on functions, applications and open issues,’. *J. Ind. Integr. Manage*, 3(4).

[71] Emiliano De Cristofaro Vitaly Shmatikov Luca Melis, Congzheng Song : Exploiting unintended feature leakage in collaborative learning. 2018.

[72] Pavel Mach et Zdenek Becvar : Mobile edge computing: A survey on architecture and computation offloading. *IEEE Communications Surveys & Tutorials*, 19(3):1628–1656, 2017.

[73] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke et Muthuramakrishnan Venkitasubramaniam : l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):3–es, 2007.

[74] Damiano Di Francesco Maesa, Paolo Mori et Laura Ricci : Blockchain based access control. *In IFIP International Conference on Distributed Applications and Interoperable Systems*, pages 206–220. Springer, 2017.

[75] Damiano Di Francesco Maesa, Laura Ricci et Paolo Mori : Distributed access control through blockchain technology. *Blockchain Engineering*, page 31, 2017.

[76] D Mahajan, S Tople et A Sharma : Domain generalization using causal matching. *arXiv:2006.07500*, 2020.

[77] Ishaan Gulrajani David Lopez-Paz Martin Arjovsky, Léon Bottou : Invariant risk minimization. 2019.

[78] Ismael Martinez, Sreya Francis et Abdelhakim Senhaji Hafid : Record and reward federated learning contributions with blockchain. *In 2019 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, pages 50–57, 2019.

[79] Matias Travizano, Martin Minnoni, Gustavo Ajzenman, Carlos Sarraute, Nicolas Della Penna : Wibson: A decentralized marketplace empowering individuals to safely monetize their personal data, cited March 2019.

[80] Thomas Ristenpart Matt Fredrikson, Somesh Jha : Model inversion attacks that exploit confidence information and basic countermeasures. 2018.

[81] B. McMahan, E. Moore, D. Ramage, S. Hampson et B.A. Arcas : Communication-efficient learning of deep networks from decentralized data. *In Artificial Intelligence and Statistics*, page 1273–1282.

[82] B. McMahan et D. Ramage : Federated learning: Collaborative machine learning without centralized training data. 2017.

[83] Brendan McMahan et Daniel Ramage : Federated learning: Collaborative machine learning without centralized training data. *Google Research Blog*, 2017.

[84] Gihan J Mendis, Moein Sabounchi, Jin Wei et Rigoberto Roche : Blockchain as a service: An autonomous, privacy preserving, decentralized architecture for deep learning. *arXiv preprint arXiv:1807.02515*, 2018.

[85] C.R. Merkle : 'method of providing digital signatures,'. *U.S. Patent*, 4:309 569,.

[86] Silvio Micali : Algorand: the efficient and democratic ledger. *arXiv preprint arXiv:1607.01341*, 2016.

[87] Amir Houmansadr Milad Nasr, Reza Shokri : Comprehensive privacy analysis of deep learning. 2018.

[88] Shuang Wang Yuhou Xia-Xiaoqian Jiang Miran Kim, Yongsoo Song : Secure logistic regression based on homomorphic encryption design and evaluation. 2018.

[89] Satoshi Nakamoto *et al.* : Bitcoin: A peer-to-peer electronic cash system. 2008.

[90] Arvind Narayanan et Vitaly Shmatikov : Robust de-anonymization of large sparse datasets. *In 2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 111–125, 2008.

[91] Nasr, Reza Shokri et Amir Houmansadr : Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. *arXiv:1812.00910*, 2018.

[92] M. Oberst, N. Thams, J. Peters et D. Sontag : Regularizing towards causal invariance: Linear models with proxies. *In ICML*.

[93] Ocean Protocol Foundation with BigchainDB GmbH and Newton Circus (DEX Pte. Ltd.) : Ocean protocol: A decentralized substrate for ai data services technical whitepaper, cited March 2019.

[94] X. Peng, Z. Huang, Y. Zhu et K. Saenko : Federated adversarial domain adaptation. arXiv preprint arXiv:1911.02054 .

[95] Brendan Avent Aurélien Bellet-Mehdi Bennis Arjun Nitin Bhagoji Keith Bonawitz Zachary Charles Graham Cormode Rachel Cummings Peter Kairouz, Brendan Mcmahan : Advances and open problems in federated learning. *In Advances and Open Problems in Federated Learning*. arxiv, 2019.

[96] J. Peters, P. Buhlmann et N. Meinshausen : Causal inference by using invariant prediction: identification and confidence intervals. *Journal of the Royal Statistical Society*, 78(5):947–1012,. Series B (Statistical Methodology),.

[97] Julien Stainer Peva Blanchard, Rachid Guerraoui : Machine learning with adversaries: Byzantine tolerant gradient descent. 2017.

[98] N. Pfister, P. Buhlmann et J. Peters : Invariant causal pre- diction for sequential data. *Journal of the American Statistical Association.*

[99] Foster J Provost et Daniel N Hennessy : Scaling up: Distributed machine learning with cooperation. *In AAAI/IAAI, Vol. 1*, pages 74–79. Citeseer, 1996.

[100] D. Puthal, N. Malik, S.P. Mohanty, E. Kougianos et C. Yang : ‘the blockchain as a decentralized security framework.

[101] T. Klein R. C. Geyer et M. Nabi : Differentially private federated learning: A client level perspective. 2017.

[102] Congzheng Song Vitaly Shmatikov Reza Shokri, Marco Stronati : Membership inference attacks against machine learning models. 2017.

[103] Moin Nabi Robin C. Geyer, Tassilo Klein : Differentially private federated learning: A client level perspective. 2017.

[104] Roger Haenni : Datum network: The decentralized data marketplace, cited March 2019.

[105] E. Rosenfeld, P. Ravikumar et A. Risteski : The risks of invariant risk minimization. arXiv preprint arXiv:2010.05761,.

[106] Elan Rosenfeld, Pradeep Ravikumar et Andrej Risteski : The risks of invariant risk minimization, 2021.

[107] M. Fredrikson S. Jh S. Yeom, I. Giacomelli : Privacy risk in machine learning: Analyzing the connection to overfitting. 2018.

[108] BLOCKCHAIN LUXEMBOURG S.A. : Average bitcoin block size, line chart, 2019.

[109] A.K. Sahu, T. Li, M. Sanjabi, M. Zaheer, A. Talwalkar et V. Smith : On the convergence of federated optimization in heterogeneous networks. *CoRR.* URL:.

[110] Lakshmi Siva Sankar, M Sindhu et M Sethumadhavan : Survey of consensus protocols on blockchain applications. *In 2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS)*, pages 1–5. IEEE, 2017.

[111] Peter Richtarik Barnabas Poczos-Alex Smola Sashank Reddi, Jakub Konecny : Aide fast and communication efficient distributed optimization. 2016.

[112] B. Schneier : *Applied Cryptography, Protocols, Algorithms and Source Code in C.* Wiley, Hoboken, NJ, USA, 2nd édition.

[113] R. Schollmeier : ‘a definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications,’. *In Proc. 1st Int. Conf*, page 101–102, Peer Peer Comput., Linkoping, Sweden.

[114] Micah J Sheller, G Anthony Reina, Brandon Edwards, Jason Martin et Spyridon Bakas : Multi-institutional deep learning modeling without sharing patient data: A feasibility study on brain tumor segmentation, 2018.

[115] Zheyan Shen, Jiashuo Liu, Yue He, Xingxuan Zhang, Renzhe Xu, Han Yu et Peng Cui : Towards out-of-distribution generalization: A survey, 2021.

[116] Aditya V. Noris Shruti Tople, Amit Sharma : Alleviating privacy attacks via causal learning. 2020.

[117] Joao Sousa, Alysson Bessani et Marko Vukolic : A byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform. *In 2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 51–58. IEEE, 2018.

[118] M. Swan : *Blockchain, Blueprint for a New Economy.* O'Reilly Media, Sebastopol, CA, USA.

[119] T. Hoefler T. Ben-Nun : Demystifying parallel and distributed deep learning: An in-depth concurrency analysis. 2018.

[120] TORSTEN HOEFLER TAL BENNUN : Demystifying parallel and distributed deep learning: An in-depth concurrency analysis. 2018.

[121] Bo Xing Tshilidzi Marwala : Blockchain and artificial intelligence. 2018.

[122] T. Tuor, S. Wang, B.J. Ko, C. Liu et K.K. Leung : Overcoming noisy and irrelevant data in federated learning. arXiv e-prints , arXiv–2001.

[123] Uchi Ugobame Uchibeke, Sara Hosseinzadeh Kassani, Kevin A Schneider et Ralph Deters : Blockchain access control ecosystem for big data security. *arXiv preprint arXiv:1810.04607*, 2018.

[124] Maziar Sanjabi Ameet Talwalkar Virginia Smith, Chao-Kai Chiang : Federated multi-task learning. 2017.

[125] M. Vukolic : 'the quest for scalable blockchain fabric: Proof-of-work vs. bft replication,'. *In Proc. Int. Workshop Open Problems Netw*, volume 29, page 112–125, Secur., Zurich, Switzerland.

[126] H. Wang, B. Ustun et F. Calmon : Repairing without retraining: Avoiding disparate impact with counterfactual distributions. *In International Conference on Machine Learning*, page 6618–6627.

[127] J. Wang, Y. Chen, W. Feng, H. Yu, M. Huang et Q. Yang : Transfer learning with dynamic distribution adaptation. *ACM Transactions on Intelligent Systems and Technology (TIST*, 11(1):1–25,.

[128] J Weng, Jian Weng, J Zhang, M Li, Y Zhang et W Luo : Deepchain: Auditable and privacy-preserving deep learning with blockchain-based incentive. *Cryptology ePrint Archive, Report 2018/679*, 2018.

[129] Gavin Wood : Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151:1–32, 2014.

[130] J. Xie, H. Tang, T. Huang, F.R. Yu, R. Xie, J. Liu et Y. Liu : 'a survey of blockchain technology applied to smart cities: Research issues and challenges,'. *IEEE Commun. Surveys Tuts*, 21(3):2794–2830,.

[131] Y. Xinyi, Z. Yi et Y. He : ‘technical characteristics and model of blockchain,’. *In Proc. 10th APCA Int. Conf. Control Soft Comput*, page 562–566. CONTROLO), Jun.

[132] R. Yang, F.R. Yu, P. Si, Z. Yang et Y. Zhang : ‘integrated blockchain and edge computing systems: A survey, some research issues and challenges,’. *IEEE Commun. Surveys Tuts*, 21(2):1508–1532,.

[133] N. Yoshida, T. Nishio, M. Morikura, K. Yamamoto et R. Yonetani : Hybrid-fl: Cooperative learning mechanism using non-iid data in wireless networks. *CoRR*. URL:.

[134] Liangzhen Lai Naveen Suda Damon Civin Vikas Chandra Yue Zhao, Meng Li : Federated learning with non-iid data. 2018.

[135] Y. Zhang, T. Xiang, T.M. Hospedales et H. Lu : Deep mutual learning. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, page 4320–4328.

[136] Y. Zhao, Y. Li, Q. Mu, B. Yang et Y. Yu : ‘secure pub-sub: Blockchainbased fair payment with reputation for reliable cyber physical systems,’. *IEEE Access*, 6: 12295–12303,.

[137] Z. Zheng, S. Xie, H. Dai, X. Chen et H. Wang : ‘an overview of blockchain technology: Architecture, consensus, and future trends,’. *In Proc. IEEE 6th Int*, page 557–564, Honolulu, HI, USA. Congr. Big Data.

[138] Dong Su Heqing Huang Jialong Zhang Tengfei Ma Dimitrios Pendarakis Ian Molloy Zhongshu Gu, Hani Jamjoom : Reaching data confidentiality and model accountability on the caltrain. 2018.

[139] Hangyu Zhu, Jinjin Xu, Shiqing Liu et Yaochu Jin : Federated learning on non-iid data: A survey, 2021.