| | |
|---|---|
| **Titre:** Title: | On Personalization of Federated Learning |
| **Auteur:** Author: | Athul Sreemathy Raj |
| **Date:** | 2023 |
| **Type:** | Mémoire ou thèse / Dissertation or Thesis |
| **Référence:** Citation: | Sreemathy Raj, A. (2023). On Personalization of Federated Learning [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/53401/ |



| | |
|---|---|
| **URL de PolyPublie:** PolyPublie URL: | https://publications.polymtl.ca/53401/ |
| **Directeurs de recherche:** Advisors: | Gabriela Nicolescu |
| **Programme:** Program: | Génie informatique |

**POLYTECHNIQUE MONTRÉAL**
affiliée à l'Université de Montréal

# On Personalization of Federated Learning

**ATHUL SREEMATHY RAJ**
Département de génie informatique et génie logiciel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie informatique

Mai 2023

**POLYTECHNIQUE MONTRÉAL**
affiliée à l'Université de Montréal

Ce mémoire intitulé :

**On Personalization of Federated Learning**

présenté par **Athul SREEMATHY RAJ**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

**Foutse KHOMH**, président
**Gabriela NICOLESCU**, membre et directrice de recherche
**Sarath CHANDAR ANBIL PARTHIPAN** , membre

## DEDICATION

*To our dad who left us, and the baby girl who arrived...*

# ACKNOWLEDGEMENTS

Foremost, I would like to convey my sincerest gratitude and acknowledgment to my director, Dr. Gabriela Nicolescu without whom I cannot imagine the successful completion of my studies. I am thankful to her for believing and entrusting me to voyage into a new field with mediocre prior knowledge.

I sincerely convey my deepest gratitude to my colleagues Irene Tenison, who started off helping fix bugs and eventually ended up guiding and co-authoring a paper with me, Kacem Khalid for contributing his time and expertise that helped shape my work into its true academic form, and Felipe Gohring de Magalhães for his thoughtful inputs and insights throughout this work and its presentations.

I thank Maroua Ben Attia and her team at Humanitas Solutions for their expert advice and weekly meetups that helped me keep track of the project.

Thanks a ton to my classmate and friend, Marc-Antoine Provost, who worked with me in building our class project during the Winter of 2022. Thanks a lot to Irina Rish and Mila for the NSL course that introduced me to the latest advancements in the field. I should also thank Compute Canada and its maintainers for providing a ground for quality research to thrive.

I also thank Humanitas Solutions, CRIAQ, IVADO, and PROMPT for funding my work and supporting me throughout the journey.

It would be a sin not to include these few names in this list: my colleague and friend Philippe Beardsell for choosing to work with me towards his goal of creating an AI-powered product; my fun and friendly lab-mate Rodolphe, Jean-Baptiste, and Oceane for the numerous parties that kept me sane; my closest amis in Montréal - Bhagya, Patrick, and Jyothis for being my go-to connections; and my wonderful roommates Emilie, Romain, Emeline, and Jérémy for being an inclusive and understanding bunch.

# RÉSUMÉ

Le monde est arrivé à un point où la technologie a non seulement connu une croissance exponentielle, mais où elle rivalise et surpasse son créateur, l'être humain. Cela a rendu la vie beaucoup plus facile pour toutes sortes d'êtres vivants à bien des égards. En se développant, la technologie est également devenue accessible à tout le monde sous la forme d'appareils intelligents, de voitures connectées, etc.. L'évolution de l'apprentissage automatique et de l'intelligence artificielle a permis aux robots de prendre en charge des tâches pénibles et dangereuses habituellement effectuées par l'homme. L'apprentissage automatique, tel que nous le connaissons aujourd'hui, utilise les informations disponibles pour apprendre et utilise ces connaissances acquises pour prédire le résultat d'un scénario qui, la plupart du temps, est inconnu auparavant. Les données utilisées pour cet apprentissage sont un facteur essentiel qui détermine la qualité des connaissances acquises et par conséquent des performances du modèle construit. Il est donc important que les données d'apprentissage soient significatives.

Tout le monde souhaite que son smartphone comprenne ses propres besoins et exigences, mais personne ne veut partager ses secrets. Il faut donc trouver une solution qui permette d'apprendre à partir des données sans laisser des tiers y accéder. L'apprentissage fédéré (FL) a pris forme pour résoudre ce défaut de sécurité associé des techniques classiques d'apprentissage automatique. FL est une méthode d'apprentissage automatique distribuée dans laquelle les données restent réparties entre les clients. Au lieu de centraliser les données, FL apprend des modèles individuels, à partir des clients du réseau, et agrège ces modèles au niveau du serveur. De cette manière, les connaissances acquises par tous les clients participants sont accessibles les uns aux autres sans partager les données.

La plupart des algorithmes d'apprentissage fédéré comme FedAVG agrègent les modèles des clients pour obtenir un modèle global. Cependant, cela conduit à une perte d'information, en particulier lorsque la distribution des données est très hétérogène entre les clients. La motivation de ce projet est une expérience simple qui prouve que les modèles globaux spécifiques aux données (où les clients sont regroupés en fonction de leur distribution de données) produisent une meilleure précision que FedAVG. Cela suggère une amélioration potentielle des performances si les clients formés sur des données similaires ont une plus grande importance dans l'agrégation des modèles. Nous utilisons des représentations de données provenant d'extracteurs de modèles de clients pour quantifier la similarité des données. Nous proposons d'utiliser une agrégation pondérée de modèles clients où le poids est calculé en fonction de la similarité des données des clients. Comme FedBABU, l'agrégation proposée basée sur la

similarité de la représentation du client est appliquée uniquement sur les extracteurs. Nous montrons empiriquement que la méthode proposée améliore les performances du modèle global dans les distributions de données hétérogènes.

# ABSTRACT

The world has come to a point where technology has not only grown exponentially but also is competing and excelling over its creator, human beings. This has made life a lot easier for all sorts of living beings in many ways. As technology grew, it also became accessible to the common man in the form of smart devices, connected cars, and henceforth. The evolution of machine learning (ML) and artificial intelligence (AI) has made it possible for robots to take over dangerous and strenuous tasks classically done by human effort. ML, as we know it today, makes use of available information to *learn* and uses that knowledge to predict the outcome of a scenario that, most of the time, is previously unknown. The data used for this *learning* is a pivotal factor that decides the quality of knowledge acquired, and hence, the performance of the learned model. Hence, it is important for the training data to be meaningful.

Everyone wishes their smartphone understands their personal needs and requirements, but nobody wants to share their secrets. This calls for a solution that learns from data without letting third parties access it. Federated learning (FL) took shape to solve this particular security flaw of classic ML techniques. FL is a distributed ML method wherein the data distributed across clients stays there. Instead of centralizing data, FL learns individual models from the clients in the network and aggregates the models at the server. In this manner, the knowledge learned from all participating clients is made accessible to each other without sharing the data.

Most federated learning algorithms like FederatedAveraging (FedAVG) aggregate client models to obtain a global model. However, this leads to a loss of information, especially when the data distribution is highly heterogeneous across clients. The motivation for this project is a simple experiment that proves that data-specific global models (where the clients are grouped based on their data distribution) produce higher accuracy over FedAVG. This suggests a potential performance improvement if clients trained on similar data have a higher importance in model aggregation. We use data representations from extractors of client models to quantify data similarity. We propose using a weighted aggregation of client models where the weight is calculated based on the similarity of client data. Like FedBABU, the proposed *client representation similarity*-based aggregation is applied only on extractors. We empirically show that the proposed method enhances global model performance in heterogeneous data distributions.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# LIST OF SYMBOLS AND ACRONYMS

| | |
|---|---|
| ML | machine learning |
| AI | artificial intelligence |
| FL | federated learning |
| FedAVG | FederatedAveraging |
| OOD | out-of-distribution |
| IoT | Internet of Things |
| IID | Independent and identically distributed |

# CHAPTER 1 INTRODUCTION

Before machine learning (ML), computer programs used to be static, and their performance stayed the same unless the code has been changed. However, the thoughts about systems that learn on their own were around way before we think they came into existence. In fact, the first-ever artificial neural network, called Stochastic Neural Analog Reinforcement Calculator (SNARC) was introduced by Marvin Lee Minsky back in 1951. Contributions from eminent personalities like Yoshua Bengio, Yann LeCun, and Geoff Hinton are often referred to as the most prominent in building modern ML techniques and thereby, the rise of artificial intelligence (AI).

## 1.1 Machine Learning and Artificial Intelligence

ML introduced the possibility of a computer program adapting and improving itself through training. ML models depend on patterns that are present in the data they are trained on, to make predictions that are extrapolations of some currently existing behavior. This ability of ML redefined how computer programs worked and opened up a wide variety of applications and solutions.

One of the key requirements for ML is the availability of meaningful data. The same affect the accuracy and usability of predictions the program makes. Various methods are used to collect this data depending on the use case. A model that helps the autopilot mode in a car would require multimedia in the form of videos and pictures, along with signals to be sent to the various components in the car to understand how certain scenarios are handled. On the other hand, a touch keyboard trying to predict the next possible word would need textual data from users. The idea is to collect real data from sources that are spread across the users and use them to understand patterns. This is where privacy becomes a concern.

## 1.2 Data Privacy in Machine Learning

Privacy has always been a fundamental right for everyone. But as technology continued to grow, data gained a lot more value and entities and enterprises all over the world started collecting data from every touch point its users interact with. The approval for this data collection was most of the time hidden and unknowingly accepted by the users in the past decade. Even though this produced better products, it also meant a lot of private data being compromised in the form of attacks by hackers, and through tech giants selling them to other

businesses for targeted marketing.

### 1.2.1 Technology in personal spaces

As technology became affordable and internet speeds grew, a lot more data started being generated in the world. From lightweight smart home devices that secured a space in our personal spaces, to high-resolution medical imagery equipment that curates multi-layer high-precision scans of vital organs; all of them produce data that could be meaningfully used to deliver a better experience to users. According to SeedScientific [2], about 2.5 quintillion ($x10^{18}$) bytes of data are produced around the world every day. This data is transferred over the network between places resulting in significant data communication overheads.

## 1.3 Distributed machine Learning

The massive volume of available data and the lack of computation power on a single workstation or a server to handle them gave way to distributed ML techniques. Distributed ML utilizes multiple nodes to train ML models, with data or models or both being split and distributed across the participating nodes. This leads to more efficient training due to the parallel nature of distributed algorithms. Even though distributed ML helps train larger models on high-volume data, this data would still need to be shipped to the nodes participating in the training process. Tapping data from consumer devices comes with its own downsides - network overhead, privacy policy infringements, potential data leaks, and henceforth. Collecting data from personal devices could also lead to reduced trustworthiness among its users; after all, nobody wants to share their secrets.

## 1.4 Federated Learning

Federated learning(FL) [3] is a distributed approach to ML, which introduced a novel method of training models without centralizing data. A global model is defined at the server, and the same is broadcast to all the participating clients, from the server. The clients train these local models individually with the local data available to the client. This means the data generated on various devices would not have to be shipped to a data center. The trained local models on clients are then sent back to the server and the server aggregates the local models to generate the new global model. These steps constitute an epoch of federated learning, and it is repeated $n$ a number of times to achieve higher accuracy. The optimization problem that federated learning tries to solve is termed federated optimization. In practical scenarios, there are a few differences when they are compared.

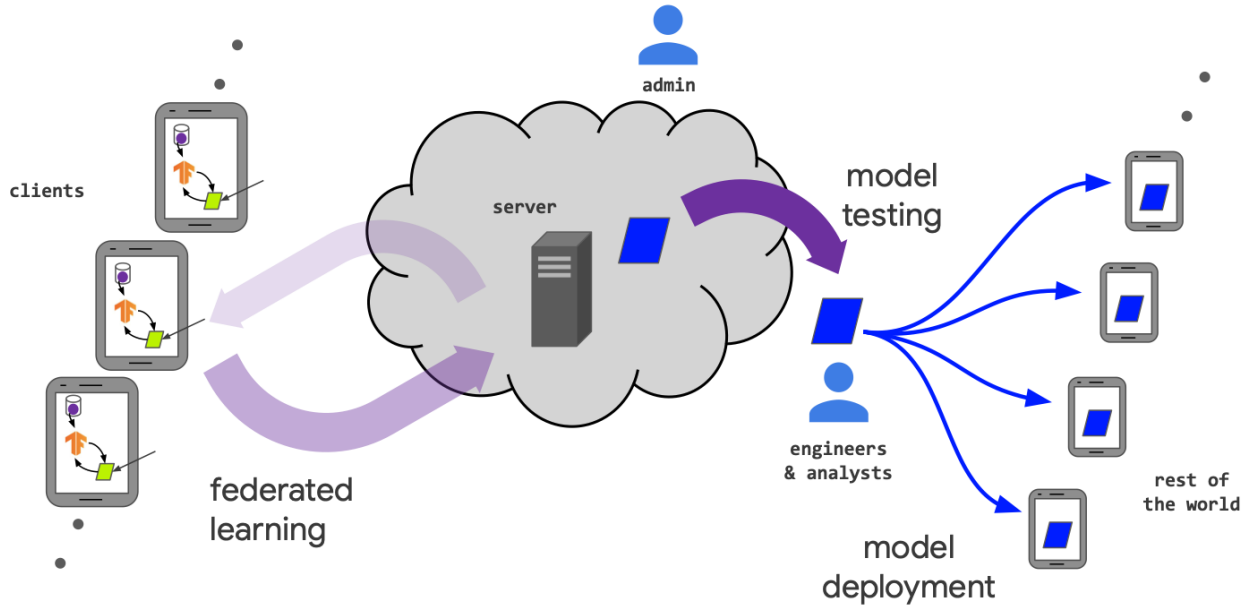


Figure 1.1 The steps involved in federated learning as given in the work [1]

### 1.4.1 Practicality of FL

The most common application of federated learning would be mobile phones and Internet of Things(IoT) devices spread across the world. Since they are personal devices used by people, there would be data-distribution skews, as well as communication constraints. These are not present in naive distributed learning methods since the participating nodes are always connected, and the data among them could be redistributed by a central entity.

#### Data distribution

Since the data on personal devices represent only one person, the user of the device, it cannot be considered to be a representation of all the participants. Hence, the data at each client can be considered to be variables that are non-independent and identically distributed *non-IID*.

#### Unbalanced

The data distribution would be diverse among the participating clients since it could be dependent on the service's usage by the client. For example, a music app trying to train a model on the client will see a lot of big variations in data availability depending on how much of a listener the user is.

#### Massively distributed

The number of clients in a classic federated learning setting is expected to be much larger compared to the average sample data point per client in the network. This is the nature of such consumer-based device networks but is also dependent on the type of devices in the network. Connected cars could be a counter-example for this statement because the number of devices would be much less and since each device is potentially collecting a lot more data the *massively distributed* nature of federated learning may not be a concern.

#### Limited communication

The devices involved in a federated learning network cannot be guaranteed to be connected to the network always. Often, only a percentage of the participating clients would be involved in training, and these clients keep changing in each epoch, especially in horizontal a federated learning setting.

### 1.4.2 Types of FL

Federated learning can be applied in different settings, across different devices and networks. Depending on the nature of the network, the data involved, and the kinds of devices in the network, federated learning is classified into different types.

#### Cross-device & Cross-silo

The entities involved in a federated learning network could be anything from a low-compute mobile device, to a high-performance server and the number of devices could go from 10s to millions.

**Cross-device** federated learning networks typically consists of mobile devices. The number of clients in these networks can be high(of the order of millions), and the data samples present in each device would be considerably less compared to the number of devices.

**Cross-silo** networks usually consist of organizations that want to perform federated training so as to share knowledge without sharing data. The number of clients in the network would usually be small, but the computing power of the devices involved would be much higher. The clients would be more controllable in this scenario, compared to the cross-device where the devices are mostly anonymous.

#### Horizontal & Vertical

Depending on the partitioning of data between clients, federated learning networks could be horizontal or vertical.

In a **horizontal** network, the dataset used by clients will have the same set of features. It could also be considered that a dataset is horizontally partitioned(by examples) and distributed across clients. Most vanilla federated learning applications would be horizontally partitioned since the data present in the clients would contain the same features but from different data points.

In a **vertical** network, the sample dataset in the clients could be having different features that belong to the same data point. This could be imagined as a dataset being split vertically and distributed among the clients. A sample application of vertically partitioned federated networks is cross-silo networks where organizations with the same user base try to profile a user, for example, a bank and a hospital containing the details about the same customer could work together to provide insights about the customer without sharing their data.

### 1.4.3 Applications of Federated learning

Since training data stays at the client, federated learning offers implicit privacy which is not guaranteed by the other ML methods. This character resulted in federated learning to be applied to data-sensitive use cases where the security of data is of utmost importance.

Federated learning is applied to personal devices like mobile phones and home assistants to train on user data that might contain highly user-specific traits. Due to the sensitive nature of such data, most users, if not all, would be apprehensive about letting this data leave their devices.

Federated learning is also applied among organizations that cannot share the data with each other but can produce better insights and personalized experiences if the knowledge could be merged. Cross-silo applications of federated learning represent such use cases.

Medical imagery devices often produce large images which may choke the network if transmitted in bulk. The limited availability of real medical images, along with the country-specific rules on data sharing makes it close to impossible to train on medical images to produce dependable results. Federated learning comes in handy in such cases to let images reside in the home country/device, and leverage the available data. Also, since the data is not transmitted, the network demands would also be low compared to a naive distributed learning network.

### 1.4.4 Performance of FL

The clients involved in a federated learning network will contain data points specific to their activity and behavior. The goal of a federated learning network is to train a model that caters to the personal behavior of these clients and at the same time provides useful predictions to a new client that joins the network.

#### Personalization in FL

Personalization of FL is the aspect of federated learning networks where the trained model caters to the behavioral pattern of any specific client involved in the training. Usually, this is achieved by training the global model on local data. This step is often referred to as the personalization of the global model.

#### Generalization in FL

Generalization of FL is the ability of a trained model to understand the invariant mechanisms among the clients to produce a model that performs adequately for a new client without personal data. This is particularly important in two scenarios:

- When a new client joins the network, the global model being sent to the client should be able to produce consumable results.
- The global models should be flexible enough to be personalized for the client, during the personalization.

## 1.5 Motivation

The core idea of this project was from a realistic use case that appears in languages where the same language is spoken differently due to the differences in slang. Even though the dialect is the same, how it's used might differ a lot depending mostly on geographical location. This thought provoked the requirement of a solution that can find out similar data points in a dataset but without matching the data points on its own. Instead, the idea was to compare how models perceive each data point and then match the ones that produce similar signals inside a network. This way, even if the data points do not look the same at the beginning, after a few transformations they might end up having similar forms.

The real-world data distributed across the clients are heterogeneous since they are dependent on user behavior and other client specifics. This is a major challenge in federated

learning since it induces client drift [4] which prevents global model convergence. Several researchers are focusing on enhancing the global model performance by effectively tackling the heterogeneity in data distribution [5–7]. Researchers have also been leveraging the data heterogeneity to enhance the personalization of the client models aiming at enhancing the client experience while the collective behavior is retained. [8–11] focus on building multiple personalized models. FedBABU [12] focuses on building a global model capable of fitting the local data of all clients. To enable this, FedBABU decouples the model into extractors and classifiers; the extractors are aggregated while the classifiers are only fine-tuned. The aggregated extractor and fine-tuned classifiers together form the local model. They show that this enhances the personalization of client models since decoupling and fine-tuning protect classifiers from learning unnecessary and irrelevant information [12].

### 1.6 Objectives

This thesis is focused on both the personalization and the generalization of ML models. The main objectives of this project along with their sub-objectives are:

Objective 1: Improve the personalization performance of ML models through a distributed communication-efficient method.

- sub-objective 1.1: Increase the accuracy of ML models on participating clients
- sub-objective 1.2: Decrease the network overhead during the training of the models

Objective 2: To build a unified generalization performance benchmarking system for ML models in an FL setting

- sub-objective 2.1: two derived datasets by inducing color and rotation skew to be used for the aforesaid benchmarking

### 1.7 Contributions

Our contributions are summarized as follows:

- We demonstrate that aggregation of all extractors as in FedBABU and other federated algorithms leads to client performance degradation and that selective aggregation based on data distribution likeliness diminishes this performance degradation.

- We propose a novel algorithm, **FedSHIBU**, which introduces a data distribution similarity-based aggregation as an alternative to aggregation of all extractors in FedBABU. A client representation similarity matrix (CRSM) is calculated to quantify the similarity across data distributions and extractors are aggregated corresponding to their respective similarity score from CRSM.
- We empirically show that similarity-based aggregation improves the personalization performance of clients in both model-decoupled setting (as in FedBABU) and full-model setting (as in FedAVG).
- We introduce two datasets called FeRotMNIST and FeCMNIST which are generated by introducing client-specific spurious mechanisms, but also without compromising the invariant nature.

# CHAPTER 2 LITERATURE REVIEW

## 2.1 Federated Learning

Federated Learning was introduced in [3]. The paper introduced FL as a communication efficient method of distributed machine learning with implicit privacy improvements compared to the existing techniques. The key advantage of federated learning as per the authors was that since data is not transferred over the network, ML models could be trained on decentralized data in a communication-efficient manner using FL. The added privacy also opened up a whole new avenue of use cases and the paper became the boilerplate project for further research in the area.

### 2.1.1 IID and non-IID dataset

A dataset being **IID** bundles two characters together:

- If the data samples show no correlation with each other, it can be considered **independent**. The most commonly expressed examples are coin tosses or die tosses. Here, each toss is independent of the previous or the next and hence every data point produced can be considered independent of each other.
- If the probability distribution of the samples is the same, across any random sample generated from the dataset, it could be called an **identically distributed** dataset. This aspect is important for federated learning because one of the key features of federated learning is that the dataset would be non-IID. This is due to the inherent nature of the data being used for most FL networks. Since it is customer data, it's very likely that the dataset-specific characters of the user which cannot be generalized across all users.

### 2.1.2 Personalization and generalization problem

To be consumable by a wider set of use cases and not be overfit, machine learning models try to discover the invariant mechanisms present in the available training dataset. This helps models perform well on datasets that were not trained on.

**Personalization**

**Generalization**

### 2.1.3 Goal of federated learning

The main goal of federated learning algorithms is to generate a global model that has the ability to produce predictions for all the clients involved in the training process. Even though data distributed across the clients in a federated setting is highly non-IID.

## 2.2 Federated Averaging - FedAVG

Instead of shipping data from individual nodes to a central server, FL trains models on each of the client nodes and then aggregates the models at a server. For aggregating the models, the authors of [3] used a simple technique by averaging the weights of each model - the authors called this method federated averaging(FedAVG).

In Algorithm 1 we assume $1, \cdots, N$ to be the set of all clients in the federated network. In every communication round or training round, a random subset of $m$ clients are chosen to participate in the training procedure. The weights of local models of all participating clients $\theta^k(0)_{i=1}^m$ are initialized with the global model parameters $\theta_G^{k-1}$, that is, $\theta_i^k(0) \leftarrow \theta_G^{k-1}$ $\forall$ i $= 1, .., m$. $\theta_i^k(0)$ is the local model parameters of client $i$ in communication round $k$ at local epoch $\tau = 0$. $\theta_G^0$ is randomly initialized for the first communication round, $k = 1$. Each client then updates its local models for $\tau$ iterations through their local data, $D_i$. The local model parameters, $\theta^k(\tau)$ from all clients are returned to the server where they are aggregated to obtain the global model $\theta_G^k$ for that communication round $k$; $\theta_G^k = \sum_{i=1}^m \theta_i^k(\tau)$. Our research focuses on a federated network with balanced data distribution or an approximately equal number of data samples per client. When the number of data samples per client varies drastically, the local model parameters are weighted in their aggregation based on their respective sample size; $\theta_G^k = \sum_{i=1}^m \frac{D_i}{\sum_{j=1}^m D_j} \theta_i^k(\tau)$ to ensure proportional contribution to the aggregation.

**Algorithm 1** FederatedAveraging [13]

**Server Executes:**

**Initialize** $w_0$
**for** each server epoch, t = 1,2,3,... **do**
  Choose $m$ clients at random
  **for** each client in C, n **do**
    $w_t^n = \text{ClientUpdate}(w_{t-1})$
    $\Delta_t^n = \frac{n_s}{\sum_{n \in C} n_s}(w_t^n - w_{t-1})$
  **end for**
  $\Delta_t = \sum_{n \in C} \Delta_t^n$
  $w_t = w_{t-1} + \eta_g * \Delta_t$
**end for**

**ClientUpdate(w):**

**Initialize** $w_0 = w$
**for** each local client iteration, i=0,1,2,3,..,n **do**
  $g_i = \nabla_{w_i} L(w_i)$
  $w_{i+1} = w_i - \eta_c \; g_i$
**end for**
**return** $w_{i+1}$ to server

### 2.3 FedBABU

FedBABU [12] is a federated learning algorithm that focuses on improving the personalization performance of FL algorithms. FedBABU considers the model in two parts - the extractor (body) and the classifier (head). The extractor of a neural network concentrates on learning abstract patterns from data, whereas the classifiers use the output of the extractors to make a decision about the class to which the input data point belongs. FedBABU aggregates only the extractors to create the global model. Classifiers are fine-tuned with local datasets available to clients to produce personalized models.

#### 2.3.1 Extractors & Classifiers

**Extractors**

Extractors are referred to the part of a neural network that finds edges and other generic features from a given input and through meaningful representations. In a typical neural

network, all the layers before the dense layer constitute the extractor. Extractors are label-unaware.

**Classifiers**

Classifier, on the other hand, accepts the output from the extractor, flatten it into a linear array, and maps the array into a potential class. Classifiers are tuned specifically to the dataset and facilitate personalized behavior.

### 2.3.2 FedBABU in heterogeneous data distribution

FedBABU averages all extractors to form a global model extractor. It then initializes the extractor of client models. The classifiers are then fine-tuned. This is particularly helpful when the data across clients are similar and the extractors retrieve features that are relevant to all clients. FedBABU performs better than FedAVG [14] when data distribution across clients is heterogeneous ($s > 5$). However, when the data is extremely heterogeneous ($s < 5$), the features learned by an extractor will be less relevant to classifiers of clients having distant data distributions. In these cases, FedBABU fails to outperform FedAVG. Heterogeneity in FedBABU [15] evaluation is limited to $s \geq 10$. From our experiments in Table 3.2, we observe that when $s < 5$ in CIFAR100, FedAVG outperforms FedBABU. However, the same was not observed in CIFAR10 which is a 10-class dataset, and the heterogeneity with $s < 5$ is not as severe as in CIFAR100 which is a 100-class dataset. We hypothesize that this is because of the presence of less relevant extractors in the aggregation which results in an extractor with diminished abilities to extract relevant features. We propose to handle this by using a weighted aggregation of extractors such that the extractors are weighted proportional to their relative relevance or similarity. We propose to quantify this relevance with a client representation similarity matrix (CRSM) and aggregate clients depending on their relevance.

### 2.3.3 FedBABU Algorithm

In FedBABU [12], client models are decoupled into classifiers (head) $\theta_{cls}$ and extractors (body) $\theta_{ext}$. After training local models at all participating clients, the extractor parameters are sent back to the server for aggregation. The extractors are aggregated leaving the classifiers unchanged $\theta^k_{G,ext} = \sum_{i=1}^{m} \frac{D_i}{\sum_{j=1}^{m} D_j} \theta^k_{i,ext}(\tau)$. Decoupling of the network to extractors and classifiers helps reduce the bias in the classifiers in settings where the data distribution varies like class-imbalance settings [15]. The classifiers $\theta^k_{G,ext}$ are fine-tuned to enhance the personalization performance of the client models. For client updates in the ClientBodyUpdate

function, the local extractor parameter $\theta^k_{i,ext}$ are updated based on the same classifier $\theta^0_{G,cls}$ such that the global parameters have the same classifier parameter as explained in Section 5.2 of [12]. It is to be noted that in FedBABU [15], all extractors are aggregated to form the global extractor which is passed over to all clients. We propose a client representation similarity matrix-based aggregation as given in the CRSMAggregation function of Algorithm 6 and further explained in Section 3.5 C.

**Algorithm 2** FedBABU [12]

**function** FEDBABU
  initialize $\theta^0_G = \{\theta^0_{G,ext}, \theta^0_{G,cls}\}$
  **for** each round $k$ = 1,...,K **do**
    $C^k \leftarrow$ random subset of $m$ clients
    **for** each client $C^k_i$ in parallel **do**
      $\theta^k_i(0) \leftarrow \theta^{k-1}_G = \{\theta^{k-1}_{G,ext}, \theta^0_{G,cls}\}$
      $\theta^k_{i,ext} \leftarrow$ **ClientBodyUpdate**$(\theta^k_i(0), \tau)$
    **end for**
$\theta^k_{i,ext} \leftarrow \sum_{i=1}^m$
  **end for**
**return** $\theta^k_G = \{\theta^K_{G,ext}, \theta^0_{G,cls}\}$

**Algorithm 3** Updating body of client

**function** CLIENTBODYUPDATE$(\theta^k_i, \tau)$
  **for** each local epoch 1,...,$\tau$ **do**
    $\theta^k_{i,ext} \leftarrow SGD(\theta^k_{i,ext}, \theta^k_{0,cls})$
  **end for**
 **return** $\theta^k_{i,ext}$

### 2.4 Other federated learning algorithms

In FedAVG [3] for each communication round, all selected $B$ fractions of clients perform $E$ local steps of the gradient descent with their local datasets. The model parameters from participating clients are averaged at the server to obtain the global model. It is equivalent to FedSGD [3] when E = 1 and each client perform stochastic gradient descent. Multiple local steps help minimize communication costs, which is a major bottleneck in FL. Quantization methods [16] and gradient descent acceleration [17] methods have been proposed to reduce communication overhead. Convergence of FedAVG under IID settings has been analyzed widely [18–20]. The convergence rate of FedAVG worsens with increasing heterogeneity

among client datasets and this has been analyzed by several works [21,22]. Multiple variations of FedAVG have been proposed to improve convergence in non-IID data distribution settings, including adding regularization to the client objective [22], normalized averaging of model parameters [6], and introducing server momentum [23]. [4] uses control variates to reduce client drift. Adaptive optimizers like Adam and Yogi have been introduced to the federated setting by [24]. Generalization in federated learning has been explored in terms of out-of-sample and participation gaps in [25]. [26] explores adaptations of IRM to exploit invariance in federated learning. Algorithms like PerFedAVG [27], Ditto [28], and FedBABU [12] focus on the personalization of clients. Differential privacy and blockchain have been used in FL to enhance data privacy.FedGMA [7] addresses the problems due to permutation variances in the neural networks.

# CHAPTER 3 ARTICLE 1 FEDSHIBU: FEDERATED SIMILARITY-BASED HEAD INDEPENDENT BODY UPDATE

Authors - Athul Sreemathy Raj, Irene Tenison, Kacem Khaled, Maroua Ben Attia, Felipe Gohring de Magalhães, Gabriela Nicolescu
Date of publication - October 20, 2022
Workshop - International Workshop on Federated Learning: Recent Advances and New Challenges in Conjunction with NeurIPS 2022 (FL-NeurIPS'22)

## 3.1 Abstract

Most federated learning algorithms like FedAVG aggregate client models to obtain a global model. However, this leads to a loss of information, especially when the data distribution is highly heterogeneous across clients. As a motivation for this paper, we first show that data-specific global models (where the clients are grouped based on their data distribution) produce higher accuracy over FedAVG. This suggests a potential performance improvement if clients trained on similar data have a higher importance in model aggregation. We use data representations from extractors of client models to quantify data similarity. We propose using a weighted aggregation of client models where the weight is calculated based on the similarity of client data. Similar to FedBABU, the proposed *client representation similarity*-based aggregation is applied only on extractors. We empirically show that the proposed method enhances global model performance in heterogeneous data distributions.

## 3.2 Introduction

Federated learning is a distributed machine learning framework where several devices collectively train a model without accessing the data distributed across these devices. The devices or clients utilize the data collected or generated at their end, to train their local models. These local models are used to effectively train a global model housed in the server without direct access to the data. These server models are then downloaded by the clients to replace their local models. The local models are further updated with their local data to continue training. The most common algorithm to train the global model in this setting is Federated Averaging (FedAVG) [14], where the trained local model parameters are averaged to form the global model. Enhancing the global model performance has been the major objective of federated learning [4, 6, 24, 29]. FedAVG and similar aggregation strategies are particularly

helpful when the distribution of data across the clients is homogeneous.

However, the real-world data distributed across the clients are heterogeneous since they are dependent on user behavior and other client specifics. This is a major challenge in federated learning since it induces client drift [4] which prevents global model convergence. Several researchers are focusing on enhancing the global model performance by effectively tackling the heterogeneity in data distribution [5–7]. Researchers have also been leveraging the data heterogeneity to enhance the personalization of the client models aiming at enhancing the client experience while the collective behavior is retained. [8–11] focus on building multiple personalized models. FedBABU [12] focuses on building a global model capable of fitting the local data of all clients. To enable this, FedBABU decouples the model into extractors and classifiers; the extractors are aggregated while the classifiers are only fine-tuned. The aggregated extractor and fine-tuned classifier together form the local model. They show that this enhances the personalization of client models since decoupling and fine-tuning protect classifiers from learning unnecessary and irrelevant information [12].

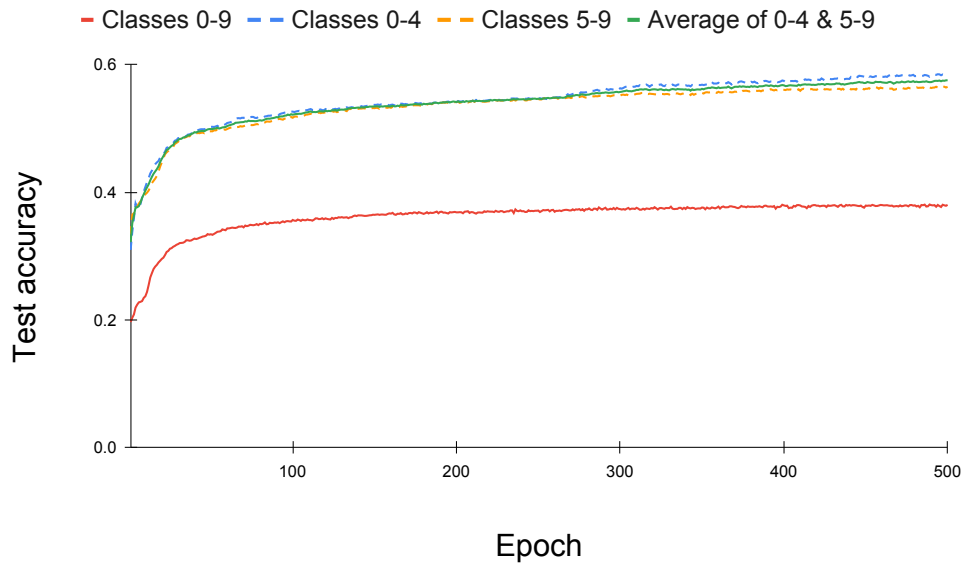


Figure 3.1 Federated Learning global models benefit from the grouping of clients based on their data distributions. The average performance of clients when their models are aggregated based on their grouping is higher than when all client models are aggregated irrespective of their data distribution.

In this paper, we focus on extractors. We investigate whether the extractors are learning

unnecessary information through aggregation and if this leads to performance degradation. To answer this question empirically, we simulate a simple federated setting with 10 clients. We distribute the CIFAR-10 dataset across these clients such that clients 0 to 4 each have examples from labels 0 to 4 and clients 5 to 9 each have examples from labels 5 to 9. In the first case, all extractors are aggregated during training. In the second case, clients 0 to 4 will have an extractor obtained by aggregating those from 0 to 4 only, and similarly, clients 5 to 9 will have an extractor obtained by aggregating those from 5 to 9 only. This is an explicit grouping of clients based on their data distribution.

We run both experiments for 500 epochs and we observe that the average performance of all clients where the extractors were grouped is better than in the case where all client extractors are aggregated. The results are shown in Figure 3.1. This indicates that aggregation of client models having dissimilar or heterogeneous data distributions diminishes model performance, unlike cases when data distribution across clients is similar or homogeneous. This decrease in performance is due to information loss with model aggregation under heterogeneous data distribution [7].

Inspired by the above observation, we propose an algorithm that aggregates the extractors based on the similarity of data distributions across clients. This similarity-based aggregation is implemented on top of FedBABU. FedBABU training involves three stages - local client model updates, extractor aggregation at the server, and classifier fine-tuning at the clients. The proposed algorithm focus on extractor aggregation. A naive aggregation of extractors of all clients in the federated network is replaced by a weighted aggregation of other client extractors. The weighting is based on the similarity between a client's data distribution with the other clients in the network. We call this FedSHIBU, **Fed**erated **S**imilarity based **H**ead **I**ndependent **B**ody **U**pdate. The proposed algorithm has been summarized in Figure 3.2. Through similarity-based aggregation, FedSHIBU prioritizes client extractors trained on similar data distributions. This ensures that irrelevant information is not injected into the extractors.

Our contributions are summarized as follows:

- We demonstrate that aggregation of all extractors as in FedBABU and other federated algorithms leads to client performance degradation and that selective aggregation based on data distribution likeliness diminishes this performance degradation.

- We propose a novel algorithm, **FedSHIBU**, which introduces a data distribution similarity-based aggregation as an alternative to aggregation of all extractors in FedBABU. A client representation similarity matrix (CRSM) is calculated to quantify the similarity across data distributions and extractors are aggregated corresponding to their respective similarity score from CRSM.
- We empirically show that similarity-based aggregation improves the personalization performance of clients in both model-decoupled setting (as in FedBABU) and full-model setting (as in FedAVG).

The remainder of the paper is organized as follows: Section 3.3 reviews the background and the state-of-the-art that relates to our work; Section 3.4 introduces the basics of federated learning and FedBABU, along with our experiment setup; we explain our algorithm, experiments and obtained results inß Section 3.5; and Section 3.6 concludes this paper.

## 3.3 Related Works

### 3.3.1 Federated Learning

FedAVG proposed in [14] aims to learn a global model from multiple local models using decentralized data. They propose averaging the local model parameters to obtain the global model. Multiple iterations at the local model enable a reduction of communication rounds required for convergence compared to FedSGD [14] where the local models are sent to the server after each iteration. However, multiple iterations cause client drift [4] when the data distribution across clients is heterogeneous. This bars the global model from converging to the optimal minima of the federated network. Several solutions have been proposed to handle this. [30] proposes balancing the data distributions using data augmentation techniques to make it closer to an IID distribution. FedDyn [31] introduces a regularization term and FedProx [5] proposes a proximal term on the local objectives so as to penalize the clients diverging from the global model. Control variates were used in SCAFFOLD [4] to minimize the drift of local models from the global model. [32] aligns the features from the client networks to improve performance. FedNova [6] normalizes the gradients before averaging gradients and FedGMA [7] masks inconsistent gradients to enhance convergence of the global model. These papers focus on improving the performance of the global model.

Personalized federated learning aims to make the local models cater to the specific data distribution at each client. Local models without federation are an alternative but every client

may not have enough data to train their respective models. [33] clusters clients using unsupervised clustering algorithms on local client updates. Clusters of similar clients would have a global model specific to that cluster. Federated multi-task learning proposes task-based global models [34]. [35] uses transfer learning to transfer knowledge across clients. Regularizers are used by [28, 36] for personalization by preventing the models from being closer to global models. PerFedAVG [27] uses bi-level optimization. FedBABU [12] decouples the models and aggregates extractors only, to limit information loss from the classifiers.

### 3.3.2 Representation Similarity Analysis

Representation Similarity Analysis (RSA) is a data-analysis framework first introduced to correlate brain activities quantitatively in neuroscience [37]. This method uses pair-wise comparisons of data to reveal more information [38]. RSA methods in neuroscience use distance matrices to compute similarity [39]. It is widely used to analyze fMRI images of the brain and its specific regions of interest [38, 40] or to differentiate between stimuli [41]. It was later adopted to quantify the relationship between deep neural networks. In transfer learning and task taxonomy, RSA was used to quantify task similarity and cluster tasks [42]. It has been used to study the evolution of networks as training progresses [43]. Various matrices like CCA [43] and CKA [44] were developed by extending this principle. However, RSA is not devoid of pitfalls. In neuroscience, when the stimuli are confounding it tends to have higher RSA scores though they are from dissimilar systems. This is called the "mimic effect" [45] and leads to false inferences.

## 3.4 Preliminaries

### 3.4.1 Federated Learning

We summarize the federated learning training procedure with notations used in Algorithm 4. Assume $1, \cdots, N$ to be the set of all clients in the federated network. In every communication round or training round, a random subset of $m$ clients are chosen to participate in the training procedure. The parameters of local models of all participating clients $\theta^k(0)_{i=1}^m$ are initialized with the global model parameters $\theta_G^{k-1}$, that is, $\theta_i^k(0) \leftarrow \theta_G^{k-1}\ \forall\ \mathrm{i} = 1, .., m$. $\theta_i^k(0)$ is the local model parameters of client $i$ in communication round $k$ at local epoch $\tau = 0$. $\theta_G^0$ is randomly initialized for the first communication round, $k = 1$. Each client then updates its local models for $\tau$ iterations through their local data, $D_i$. The local model parameters,

$\theta^k(\tau)$ from all clients are returned to the server where they are aggregated to obtain the global model $\theta_G^k$ for that communication round $k$; $\theta_G^k = \sum_{i=1}^{m} \theta_i^k(\tau)$. Our research focuses on a federated network with balanced data distribution or an approximately equal number of data samples per client. When the number of data samples per client varies drastically, the local model parameters are weighted in their aggregation based on their respective sample size; $\theta_G^k = \sum_{i=1}^{m} \frac{D_i}{\sum_{j=1}^{m} D_j} \theta_i^k(\tau)$ to ensure proportional contribution to the aggregation.

### 3.4.2 FedBABU

In FedBABU [12], client models are decoupled into classifiers (head) $\theta_{cls}$ and extractors (body) $\theta_{ext}$. After training of local models at all participating clients, the extractor parameters are sent back to the server for aggregation. The extractors are aggregated leaving the classifiers unchanged $\theta_{G,ext}^k = \sum_{i=1}^{m} \frac{D_i}{\sum_{j=1}^{m} D_j} \theta_{i,ext}^k(\tau)$. Decoupling of the network to extractors and classifiers helps reduce the bias in the classifiers in settings where the data distribution varies like class-imbalance settings [15]. The classifiers $\theta_{G,ext}^k$ are fine-tuned to enhance the personalization performance of the client models. For client updates in the ClientBodyUpdate function, the local extractor parameter $\theta_{i,ext}^k$ are updated based on the same classifier $\theta_{G,cls}^0$ such that the global parameters have the same classifier parameter as explained in Section 5.2 of [12]. It is to be noted that in FedBABU [15], all extractors are aggregated to form the global extractor which is passed over to all clients. We propose a client representation similarity matrix-based aggregation as given in the CRSMAggregation function of Algorithm 6 and further explained in Section 3.5 C.

### 3.4.3 Experimental Setup and Evaluation Criteria

ResNet18 has been used on CIFAR10 and CIFAR100 for all experiments. The separation of extractors and classifiers has been defined similarly to that in [15]. All convolutional layers including the pool layers in between form the extractor. That is, the representation returned in Algorithm 5 is the output of the last convolutional layer in the network. All dense layers following the extractor form the classifier. Heterogeneous distribution of data across clients also followed the pattern of [15] and [14]. $m$ is the number of participating clients in each communication round and we assume $m = N$ in our experiments. $s$ is the shards per user [14] and it determines the level of heterogeneity. A lower value of $s$ implies a higher heterogeneity in the data distribution. $\tau$ is the number of local epochs during client model training. A learning rate of 0.1 is used and it is decayed as in [15].

### 3.5 FedSHIBU - Federated Similarity-based Head Independent Body Update

Inspired by the success of RSA in identifying correlations between neural responses in computational neuroscience [37], [46] and in the selection of tasks for transfer learning [42], we propose FedSHIBU, where we use RSA to identify similar clients in the federated network and effectively utilize them to improve personalization performance of the clients. FedSHIBU builds upon FedBABU [12] to enhance the personalization of clients, by utilizing a data representation similarity matrix (inspired by representation dissimilarity matrix [42]) to weigh the clients based on their relative similarity.

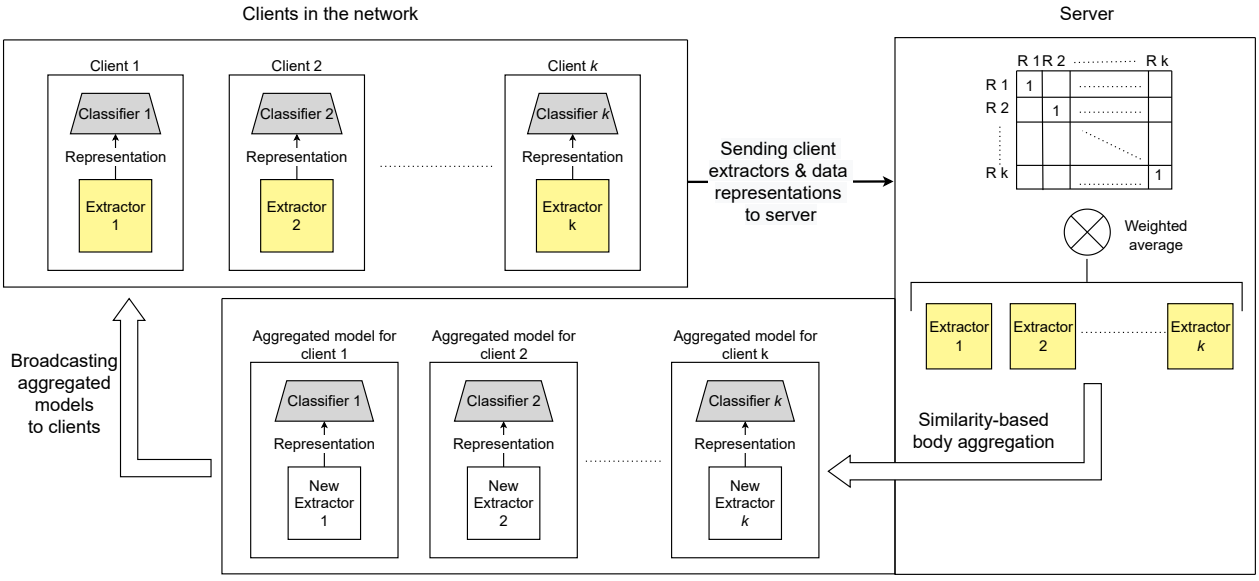


Figure 3.2 FedSHIBU - Extractors from individual clients are sent to the server, where the weighted average of models occurs. Afterward, the server broadcasts the updated models back to all participating clients.

#### 3.5.1 FedBABU in heterogeneous data distribution

FedBABU averages all extractors to form a global model extractor. It then initializes the extractor of client models. The classifiers are then fine-tuned. This is particularly helpful when the data across clients are similar and the extractors retrieve features that are relevant to all clients. FedBABU performs better than FedAVG [14] when data distribution across clients is heterogeneous ($s > 5$). However, when the data is extremely heterogeneous ($s < 5$),

**Algorithm 4** FedSHIBU - adapted from FedBABU [12]

**function** FEDSHIBU
  initialize $\theta_G^0 = \{\theta_{G,ext}^0, \theta_{G,cls}^0\}$
  **for** each round $k$ = 1,...,K **do**
    $C^k \leftarrow$ random subset of $m$ clients
    **for** each client $C_i^k$ in parallel **do**
      $\theta_i^k(0) \leftarrow \theta_G^{k-1} = \{\theta_{G,ext}^{k-1}, \theta_{G,cls}^0\}$
      $\theta_{i,ext}^k, \hat{D}_i \leftarrow$ **ClientBodyUpdate**$(\theta_i^k(0), \tau)$
    **end for**
    $\theta_{G,ext,\{1,2,..m\}}^k \leftarrow$ **CRSMAggregation**$(\theta_{ext}^k, \hat{D})$
  **end for**
**return** $\theta_G^k = \{\theta_{G,ext}^K, \theta_{G,cls}^0\}$

**Algorithm 5** Updating body of client

**function** CLIENTBODYUPDATE$(\theta_i^k, \tau)$
  **for** each local epoch 1,...,$\tau$ **do**
    $\theta_{i,ext}^k \leftarrow SGD(\theta_{i,ext}^k, \theta_{0,cls}^k)$
  **end for**
  $D_i \leftarrow$ random subset of data samples at client i
  $\hat{D}_i \leftarrow f(\theta_{i,ext}^k, D)$
**return** $\theta_{i,ext}^k, \hat{D}_i$

the features learned by an extractor will be less relevant to classifiers of clients having distant data distributions. In these cases, FedBABU fails to outperform FedAVG. Heterogeneity in FedBABU [15] evaluation is limited to $s \geq 10$. From our experiments in Table 3.1 and Table 3.2, we observe that when $s < 5$ in CIFAR100, FedAVG outperforms FedBABU. However, the same was not observed in CIFAR10 which is a 10-class dataset, and the heterogeneity with $s < 5$ is not as severe as in CIFAR100 which is a 100-class dataset. We hypothesise that this because of the presence of less-relevant extractors in the aggregation which results in an extractor with diminished abilities to extract relevant features. We propose to handle this by using a weighted aggregation of extractors such that the extractors are weighted proportional to their relative relevance or similarity. We propose to quantify this relevance with a client representation similarity matrix (CRSM) and aggregate clients depending on their relevance.

### 3.5.2 Client Similarity

Representation Similarity Analysis is a widely used methodology in neuroimaging. In [37], Representation Dissimilarity Matrix (RDM) constitutes of (1 − Pearson's correlation) of the

**Algorithm 6** Client Representation Similarity Matrix (CRSM) based Aggregation of extractors

**function** CRSMAGGREGATION($\theta^k_{ext}, \hat{D}$)
  **for** each client, $C_{cur}$=1,...,m **do**
    **for** each client, $C_{rel}$=1,...,m **do**
      $CRSM[C_{cur}, C_{rel}] \leftarrow$ **SimMet**$(\hat{D}_{C_{cur}}, \hat{D}_{C_{rel}})$
    **end for**
    $\theta^k_{G,ext,C_{cur}} \leftarrow \sum_{C_{rel}=1}^{m} \frac{CRSM[C_{cur},C_{rel}]}{\sum CRSM[C_{cur}]} \times \theta^k_{ext,C_{rel}}$
  **end for**
**return** $\theta^k_{G,ext,1,..,m}$

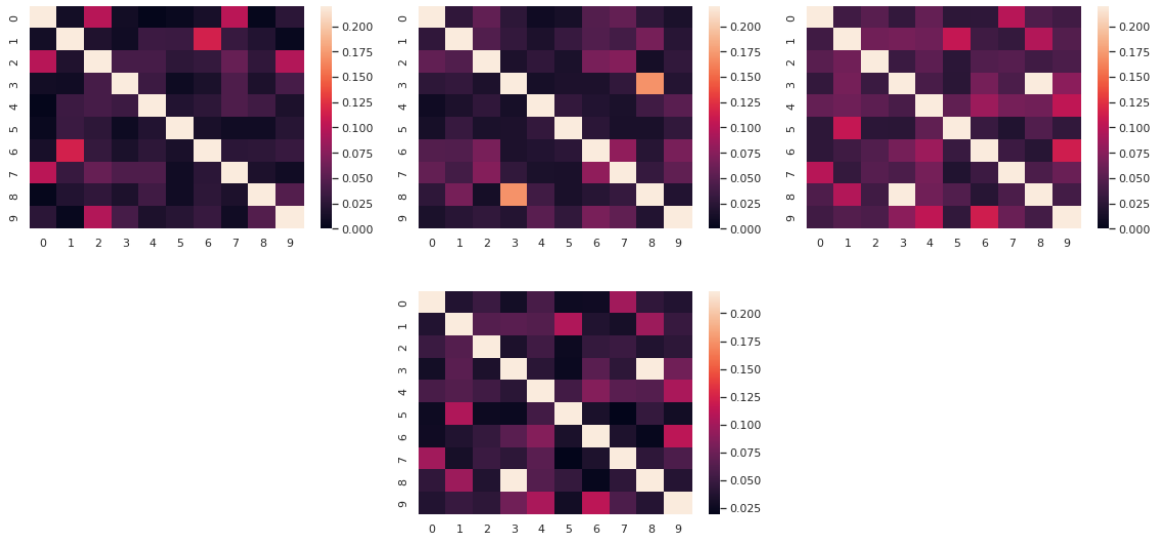

Figure 3.3 Similarity matrix from various phases of training using the proposed FedSHIBU algorithm. As training progresses we observe emerging patterns and groups and as it converges the groups become more prominent.

pairwise conditions. In computer vision transfer learning [42], Spearman's correlation was used on these matrices to compute the similarity score of two DNNs. This was because RDM cannot be used directly since the comparison is across tasks and the shapes of representations across tasks will be different. However, in our federated scenario, all client models are assumed to have the same architecture. Hence the shape of representations from extractors is expected to be the same. To quantify the relationship between clients in federated learning, we propose using a client representation similarity matrix (CRSM). CRSM is a square matrix of size $N$x$N$, where each element is a pairwise similarity score of neural network representations from client extractors. $CRSM[C_{cur}, C_{rel}]$ is the similarity of the current client $C_{cur}$ to another relative client $C_{rel}$. To measure the similarity score of neural network representations

we use CKA linear and CKA RBF [44].

CRSM at various levels of training is shown in Figure 3.3. The matrices are from 10 client federated networks where the data distribution is heterogeneous ($s = 5$). We observe that intially the grouping of clients changes very frequently. As training progresses and nears convergence, patterns are formed. The sub-figures in Figure 3.3 are from rounds 0, 150, 250, and 300 respectively. Until round 150, the values in CRSM varies drastically and in round 150 client 3 and 8 identifies each other as relatively similar to each other. As training progresses and reaches round 200, the relevance is further strengthened and almost all clients have identified their relatives. With further training towards round 300, the score of similar clients further increases and that of less-similar client decreases. That is, the gap between the groups are widened making the groups more prominent.

Table 3.1 Test Accuracy of FedSHIBU, FedBABU, and FedAVG on CIFAR10 distributed across 100 clients with full participation in each round. The algorithms are compared across varying data heterogeneity where s (shards per user) = 2 implies extreme heterogeneity and s = 100 implies homogeneous data distribution.

| **s** | **FedSHIBU (CKA Linear)** | **FedSHIBU (CKA RBF)** | **FedBABU** | **FedAVG** |
|---|---|---|---|---|
| 2 | **94.04** | 90.69 | 68.43 | 58.13 |
| 3 | **81.12** | 78.62 | 77.74 | 69.58 |
| 4 | 85.07 | **85.9** | 84.06 | 83.88 |
| 5 | **89.26** | 88.18 | 86.93 | 85.96 |
| 8 | 89.32 | **89.92** | 89.38 | 89.01 |
| 10 | **90.34** | 90.28 | 90.24 | 89.67 |
| 20 | 90.64 | **90.93** | 90.85 | 90.58 |
| 50 | 90.88 | 90.94 | 91.03 | **91.28** |
| 100 | 90.93 | **91.23** | 90.87 | 90.94 |

### 3.5.3 FedSHIBU algorithm

Based on the discussion in the above subsections, we propose a new FL algorithm called FedSHIBU (**Fed**erated **S**imilarity based **H**ead **I**ndependent **B**ody **U**pdate), an improvement upon the decoupled federated training strategy introduced in FedBABU [12]. By decoupling extractors and classifiers, only the body is trained while the head is never trained. This

enhances the performance of client models since extractors hold information on data representations that are useful for all clients. Aggregating them enhances client performance. The classifiers hold information related to linear decision boundaries of clients and these are client data specific. Aggregating these leads to performance degradation. The algorithm and implementation are explained in detail in section 5.2 of [12].

Table 3.2 Test Accuracy of FedSHIBU, FedBABU, and FedAVG on CIFAR100 distributed across 100 clients with full participation in each round. The algorithms are compared across varying data heterogeneity where s (shards per user) = 2 implies extreme heterogeneity and s = 100 implies homogeneous data distribution.

| **s** | **FedSHIBU (CKA Linear)** | **FedSHIBU (CKA RBF)** | **FedBABU** | **FedAVG** |
|---|---|---|---|---|
| 2 | **91.12** | 41.65 | 15.55 | 24.92 |
| 3 | **54.12** | 41.95 | 33.03 | 37.98 |
| 4 | **57.96** | 49.47 | 47.22 | 47.27 |
| 5 | **59.48** | 53.62 | 53.88 | 52.36 |
| 8 | **63.63** | 62.71 | 60.77 | 60.36 |
| 10 | **69.72** | 66.07 | 60.46 | 62.45 |
| 20 | **72.33** | 69.97 | 71.86 | 67.31 |
| 50 | **73.88** | 70.93 | 73.44 | 69.01 |
| 100 | 72.97 | 71.31 | **73.37** | 70.17 |

Unlike FedBABU, extractors for each client in FedSHIBU are calculated at the server with a weighted aggregation of other client extractors in the network, where the weighting is dependent on the data representation similarities across clients in the network. For the same, FedSHIBU requires client data representations to be sent to the server besides client updates. A fixed number of data samples are randomly chosen at each client and their representations form the extractors are retrieved after local training of the client. Additional privacy mechanisms can be introduced to these representations to further enhance privacy of the clients. The representations from clients are used to quantify the similarity of data across the clients in the network using matrices like CKA, CCA and so on. The scores are collected as a matrix to from CRSM. A client extractor is obtained by aggregating other client extractors in the network with a weighting corresponding to their score in the matrix. This ensures that similar clients, having useful features gets weighed more in the agggregation than dissimilar clients, whose features may be less useful for the client model under consideration.

The training procedure of FedSHIBU is described in Algorithm 4. FedSHIBU introduces **CRSMAggregation** as given in Algorithm 6 as an alternative to naive aggregation of all extractors in FedBABU. CRSMAggregation function requires all client extractor parameters $\theta^k_{ext,1,2,..m}$ (also represented as $\theta^k_{ext}$) and all client representations from their extractors $\hat{D}_{1,2,..m}$ (also represented as $\hat{D}$). It calculates the similarity of representations of all clients to each other as a proxy of the similarity of data distributions at the respective clients. **SimMet** in Algorithm 6 can be any similarity metric. The similarities are compiled into a matrix - CRSM. The aggregated extractor for each client $C_{cur}$ is calculated by aggregating the client extractors by weighting them proportional to their similarity to other clients $C_{rel}$ in the federated network. This similarity is quantified in the corresponding row of CRSM $CRSM[C_{cur}]$ as given in Algorithm 6. It is to be noted that each client's aggregated extractors would differ since the weightage of different clients would vary in each aggregation.

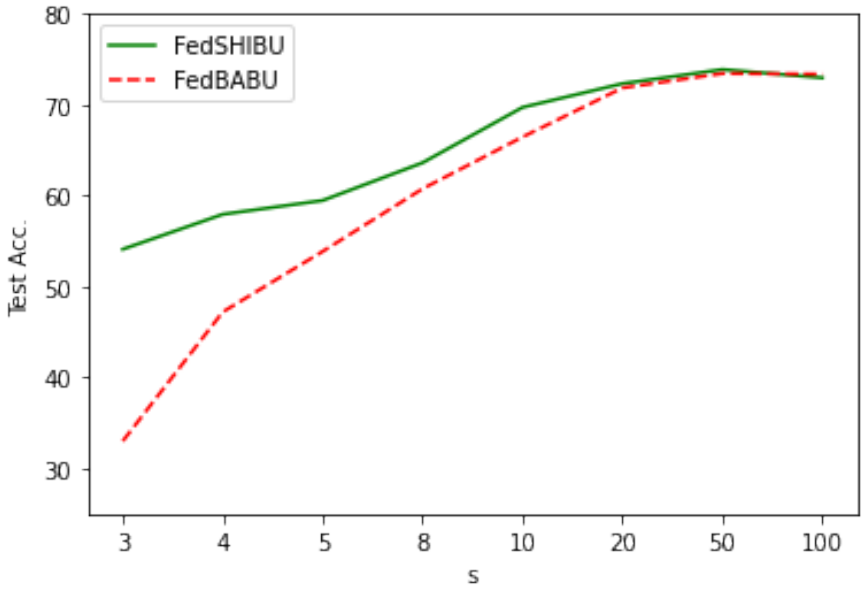


Figure 3.4 FedSHIBU outperforms FedBABU when data is extremely heterogeneous. With decreasing heterogeneity, the difference in performance decreases.

FedSHIBU aims to handle extreme data heterogeneity, where FedBABU fails as observed in Table 3.1 and Table 3.2. Figure 3.4 plots the test accuracy of FedSHIBU and FedBABU across varying heterogeneity represented by $s$. We observe that when the data is heterogeneous, FedSHIBU outperforms FedBABU, and with increasing homogeneity, the difference between the accuracy of FedSHIBU and FedBABU decreases. That is, FedSHIBU is signif-

icantly better under extreme heterogeneity. This supports our hypothesis that FedSHIBU is more capable of handling information loss than FedBABU. Furthermore, when the data distribution is homogeneous, all clients would be weighed equally and FedSHIBU would equal FedBABU.

**Algorithm 7** FedAVG with CRSM Aggregation

initialize $\theta_G^0 = \{\theta_{G,ext}^0, \theta_{G,cls}^0\}$
**for** each round $k$ = 1,...,K **do**
  $C^k \leftarrow$ random subset of m clients
  **for** each client $C_i^k$ in parallel **do**
    $\theta_i^k(0) \leftarrow \theta_G^{k-1} = \{\theta_{G,ext}^{k-1}, \theta_{G,cls}^0\}$
    $\theta_i^k, \hat{D}_i \leftarrow$ **ClientBodyUpdate**$(\theta_i^k(0), \tau)$
  **end for**
  $\theta_{G,\{1,2,..m\}}^k \leftarrow$ **CRSMAggregation**$(\theta^k, \hat{D})$
**end for**
**return** $\theta_G^k$
**function** ClientBodyUpdate$(\theta_i^k, \tau)$
  **for** each local epoch 1,...,$\tau$ **do**
    $\theta_i^k \leftarrow SGD(\theta_i^k)$
  **end for**
  $D_i \leftarrow$ random subset of data samples at client i
  $\hat{D}_i \leftarrow f(\theta_{i,ext}^k, D)$
**return** $\theta_i^k, \hat{D}_i$

### 3.5.4 Similarity-based aggregation in FedAVG

FedSHIBU proposes using a similarity-based aggregation on top of FedBABU to reduce information loss with heterogeneous data. FedBABU handles information loss from classifiers [12] and we hypothesize that the similarity-based aggregation reduces that from the extractors. To further understand the effect of representation similarity-based aggregation, we apply CRSMAggregation on full federated averaging without model decoupling which was introduced by FedBABU. This helps understand the performance improvement contributed by similarity-based aggregation specifically. Figure 3.4 plots test accuracies of FedAVG and FedAVG with similarity-based aggregation on heterogeneous data ($s = 5$). We observe significant performance improvement along with faster convergence when similarity-based aggregation is employed. This further validates the claim of using similarity-based aggregation in federated learning. FedAVG with similarity-based aggregation is given in the Algorithm 7.

### 3.6 Conclusion

This work is an improvement of the existing FedBABU [12] algorithm, by implementing an intelligent selection of clients into producing personalized models for the clients. The similarity-based aggregation of client models yields better performance than naive averaging and the existing FedBABU algorithm. FedSHIBU converges faster than FedBABU in terms of number of epochs and shows higher accuracy on the client-specific datasets. However, the calculation of the similarity matrix is an overhead taken by the central node and costs compute. But since this activity is on a device with adequate resources, this wouldn't be a limiting factor in real-life scenarios. The generalization performance of FedSHIBU should be considered outside the scope of this work, and hence, is not bench-marked in this paper. It is true that this algorithm deviates from the core ideas of FL to have a global model. Again, since the target of this project is to improve personalization performance, and not generalization, it could be considered out of the scope of this work. In the future, we would like to experiment with more metrics to quantify client data similarity. We would also like to investigate the performance improvements of FedSHIBU in more complex datasets and distribution skews to further back the claims of this paper.

# CHAPTER 4 FEDERATED LEARNING BENCHMARK FOR DOMAIN GENERALIZATION

## 4.1 Abstract

Out-of-distribution (OOD) generalization has always been the major scale on which the abilities of novel ideas are being benchmarked, in the field of machine learning. New models are being tested and pushed their limits on how they perform in a scenario that was previously not seen, so as to understand their capabilities in producing human characters. Everybody wishes their virtual assistant understood their personality well, but nobody wants their secrets to be at risk. In today's world where privacy is of utmost priority, Federated Learning (FL) [13] opened a whole new arena of possibilities by introducing a method to learn insights from data, without needing the data to be shared. Training a global model that can produce personalized results in participating clients was one of the key contributions of federated learning. FL works in a distributed manner, by training models on participating clients, and then aggregating them at a central server. The participating clients are usually high in number and have low-compute capabilities. Edge devices like mobile phones, home assistants, and drones benefit from this technique. Since its inception, scientists across the world have been keenly investigating different methods to improve the performance of FL. DomainBed is a well-established benchmark for the OOD performance of a bunch of algorithms. The repository is open-source and is actively maintained by the community. This work introduced a new repository containing experiments to benchmark the performance of OOD generalization algorithms in a federated learning setup. The algorithms being tested are those hand-picked from DomainBed and are applied on the client side in a federated setting. Existing works address the issue of the OOD generalization by introducing changes to the aggregation methods used to amalgamate client models into global models. We define two datasets extended from MNIST, to cater to our need for a non-IID dataset in a federated setting. The distribution shift is introduced by skewing features in the existing dataset. Color is introduced as a label-independent feature, and so is rotation. We establish through our experiments that among the existing loss calculations, empirical risk minimization (ERM) produces the highest accuracy in an OOD dataset.

## 4.2 Introduction

The growing accessibility and computational power of edge devices, such as consumer smartphones, personal computers, and smart home devices, coupled with the increasing privacy concerns about the vast amount of data generated each day have enabled a new distributed machine learning paradigm. Federated Learning (FL) is a decentralized training approach that enables training statistical models directly on remote devices, thus overcoming the challenges of data silos and data sensibility [13] by holding the data where they are generated and not sharing them to any central server or storage unit. However, training in heterogeneous and massive networks introduces novel challenges that require a fundamental departure from standard approaches in large-scale machine learning. One of these challenges is statistical heterogeneity. In fact, devices generally transgress the independent and identically distributed (IID) data generation assumptions. Indeed, the number of data points across devices may significantly vary in their number and distribution. As an example, when training a next-word prediction model on a user's smartphone typing data, clients located in geographically distinct regions generate data from different distributions, but there exists enough commonality between the distributions that we may still want to train a single model.

Therefore, in practical federated learning, one of the major objectives is to train a global model that improves its performance on non-participating clients. Hence, we can draw a parallel between federated learning and domain generalization algorithms that tackle the problem of OOD generalization. While there exists a vast collection of OOD benchmarks, few OOD generalization benchmarks exist for federated learning. In this paper, we propose a new extension of DOMAINBED [47], a testbed for the domain generalization, to a federated learning setting.

FL has proven itself to be a greener [48] method of machine learning because of its low footprint, and ability to be applied on resource-constrained devices. The privacy-preserving [49] character of FL makes it an adaptable solution for organizations like "Enhancing Neuroimaging Genetics through Meta-Analysis" (ENIGMA), which is currently looking for methods to share data following existing GDPR guidelines. Since data stays at the source, FL involves lesser network requirements [50] compared to the conventional distributed machine learning techniques. All these have attracted research groups and organizations from around the world, into refining and redefining the notions in FL. In this project, we study the effect of various OOD generalization techniques in FL and come up with a unified benchmark for the same.

Our contributions are:

- A reusable client-level feature-skewed dataset extended from MNIST, in both 1-channel and 3-channel versions. We call them FeCMNIST and FeRotMNIST.
- An open-sourced repository built to collect and compare out-of-generalization distribution algorithms and benchmark them in a federated learning setup, on the client's side (https://github.com/athuljayaraj/FeDomainBed)
- We establish through the experiments that implementing most current OOD algorithms on the client side do not effectively enhance generalization performance in federated learning.

## 4.3 Background

In this section, we lay the background for this work with related works covering OOD generalization benchmarks, existing algorithms, and the federated learning paradigm.

### 4.3.1 OOD Problem definition

Domain generalization [51] is an extension of the traditional supervised learning task where training datasets from multiple domains are available to train our predictor. Each domain $d$ is characterized by a dataset $D^d = (x_i^d, y_i^d)_{d=1}^{n_d}$ containing IID examples from some probability distribution $P(X^d, Y^d)$, for all training domains $d \in \{1, ..., d_{tr}\}$. The goal of out-of-domain generalization is to learn a predictor able to perform well on unseen data from related yet different test domain $d_{te} = d_{tr} + 1$. As none of test data is available during training, we must assume the existence of some statistical invariances across training and testing domain in order to incorporate some of these invariances into the predictor while some spurious mechanisms exists to differentiate the domains or environments.

### 4.3.2 OOD generalization algorithms

During the last decade, several algorithms were developed to be able to generalize to OOD. Most of these algorithms incorporate invariances to create predictors that rely on the causes of the label to make predictions. For instance, Invariant Risk Minimization (**IRM**) [51] learns a feature representation $\phi(X^d)$ such that the optimal linear classifier on top of that representation matches across domains, Variance-Risk Extrapolation (**V-REx**) [52] performs robust learning over a perturbation set of extrapolated domains and penalizes the variance of training risks. [53] introduces a binary **AND-mask** to mask out gradient updates of parameters that are inconsistent across environments. A smoothened version of this mask (**SAND**

**mask**) was introduced in [54]. [55] uses information bottleneck principles alongside invariance principles to generalize better to OOD test domains. There are more OOD algorithms that aim to generalize to OOD test domains as mentioned in [47], which benchmarks the various OOD algorithms in centralized settings on a variety of datasets, models, and experiment settings.

### 4.3.3 Existing benchmarks for OOD generalization

This section focuses on the current state of OOD benchmarks.

#### Synthetic datasets

Some synthetic datasets were created to simulate distributional shifts and gain a better understanding of generalization failures in deep learning. The most typical ones, including ImageNet [56] variants (e.g. ImageNet-A, ImageNet-C, ImageNet-R) adopt special data selection or perturbation to generate testing data with distributional shifts. Others, from MNIST variants (e.g., Colored MNIST, MNIST-R), simulate different environments by coloring or rotating original images. These datasets are useful for the preliminary study and verification of OOD generalization algorithms.

#### Image datasets

Many non-synthetic datasets were proposed, some with naturally occurring distribution shifts and some with artificially induced distribution shifts. Widely used in domain generalization, several of these datasets are composed of different renditions of the same labels, e.g., PACS [57], DomainNet [58], and Office-Home [59]. Other datasets have strong spurious features that create shortcuts to minimize the empirical risk, e.g., Waterbirds [60] which contains birds' images with either water or land backgrounds, and NICO [61] which elaborately selects visual contexts with various types, including background, attribute, and etc.

#### Benchmarks

Over the years, due to the observed inconsistencies in experimental conditions (e.g., datasets, architectures, and model selection criteria) rendering realistic comparisons difficult, researchers have proposed a number of benchmarks to facilitate fair OOD comparisons. Some of these benchmarks, such as DOMAINBED [62] and OOD-Bench, focus on static computer vision tasks while providing a myriad of algorithms and datasets. Others, like WOODS [63], extend

the work done by covering a diverse range of data modalities stemming from time series tasks, such as videos, brain recordings, and sensor signals.

**Federated Learning Training**

Federated learning is an emerging privacy-preserving distributed machine learning approach where a shared model is built by performing distributed training locally on participating devices (clients) and aggregating the local models into a shared one. The traditional template orchestrated by a server (service provider) for FL training is as follows:

1. **Client selection:** The server samples a number of clients from the set of eligible clients. For example, the server might only consider phones that have an unmetered Wi-Fi connection.
2. **Broadcast:** The selected clients download the model weights and a training framework from the server.
3. **Client computation:** Each client locally computes an update to the model by executing the optimization algorithm, e.g., Stochastic Gradient Descent, on the local data.
4. **Aggregation:** The server collects an aggregate of the devices updates. This stage can also include other techniques such as noise addition and clipping for differential privacy.
5. **Model update:** The server locally updates the shared model based on the aggregated update computed from the participating clients.

### 4.3.4 Federated Learning algorithms for non-IID data samples

While federated learning has emerged as an attractive distributed learning paradigm, most of the existing algorithms ignore the fact that the data on each client node are collected in a non-IID manner due to the varying client behavior. In fact, most statistical learning algorithms used in federated learning strongly rely on the IID assumption on client data. However, in practice, data distribution shift issues between nodes are common. Recently, some approaches were proposed to tackle these issues and enhance the OOD accuracy of the final learned model. One of these techniques, CausalFed [26], collaboratively learns causal features common to all participating clients, which enhances OOD robustness in the federated learning setting. Another method, FedGMA [7], inspired by the invariance principles in [53], proposes a gradient-masked averaging approach for federated learning as an alternative to the standard averaging of client updates. This, in turn, learns a model that is locally optimal across different simultaneous clients.

## 4.4 Connection of FL to OOD Generalization

OOD generalization is often formalized using the notion of domains or environments (IRM paper citation). These could represent different measuring circumstances, locations, times, experimental conditions, contexts, etc. Typically, environments share some invariant mechanisms. Nevertheless, they can possess spurious features that differ across environments. The concept of environments can be extended to federated learning by considering each client as producing a set of data generated from a different environment. In the federated learning setting, all clients share some underlying invariant mechanism in order to train a global model. Each client also has its own spurious mechanisms or data distribution. As an example, consider the scenario of different clients, corresponding to smartphone users taking pictures of dogs to build a dog classifier. Each user may have a different camera and may take pictures of different subsets of dogs. Thus, the clients may differ in terms of the label distribution and camera-related image characteristics, corresponding to spurious mechanisms. However, all clients share the same invariant mechanism, which in this case is the overall set of dog breeds. While the goal of OOD generalization is to improve the performance of a model on a data distributions that are related yet different from the training distribution, one of the major objectives of practical FL is to train a global model that improve its performance on non-participating clients. Hence, one way to frame the goal for FL global models is to enhance the performance across a large set of related clients that may have different data distributions.

## 4.5 Methodology

### 4.5.1 Repository

At the heart of our experiment is FeDomainBed, a PyTorch-based testbed that extends DomainBed [47] to the federated learning setting. The initial release includes seven algorithms, two datasets, and one aggregation method. Contributions via pull requests from fellow researchers are welcomed.

### 4.5.2 Datasets

FeDomainBed includes downloaders and loaders for two multi-domain images classification tasks: Colored MNIST(CMNIST) [51] and Rotated MNIST(RotMNIST). Data has been skewed in two different ways to generate non-IID distribution across clients:
(i) Label distribution skew Label distribution skew was introduced by dividing the dataset

into shards of images from same labels, and then distributing them across clients in such a way that no client gets more than 2-3 shards. Since the number of shards per client is limited to a small number, we can guarantee that no clients would get data from all the 10 labels, and hence keeps the distribution non-IID.

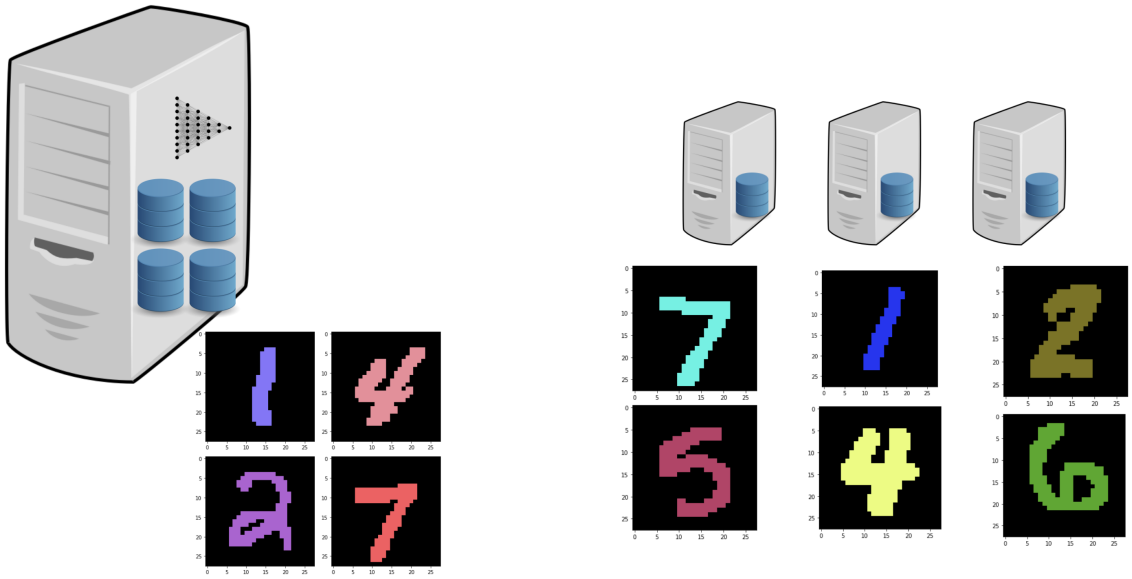

Figure 4.1 Federated version of MNIST with induced feature skew by varying the distribution of colors among classes and clients.

(ii) Feature distribution skew Depending on the dataset, we have introduced two new features to the existing single-channel MNIST dataset: colors and rotation. A skew based on these features were established by randomly altering the images and then distributing them across the clients. The feature staggering is controllable through arguments. In ColorMNIST, the number of colors per class is chosen through parameters. Once done, a random set of colors is chosen for each client, and the test set is created with a held-out set of colors. For RotMNIST, instead of the number of colors, the angle of rotation around which rotation is imparted is configurable.

### 4.5.3 Algorithms

Studies have already established a few OOD generalization algorithms in federated learning. CausalFed-IRM and FedGMA are two major works in this arena. CausalFed-IRM implements a split-learning method spread across the participating clients, and the central server, and calculates the using IRM. FedGMA assigns weights to the gradients in such a way that the updates shown by the most number of clients get a higher importance than others. Both

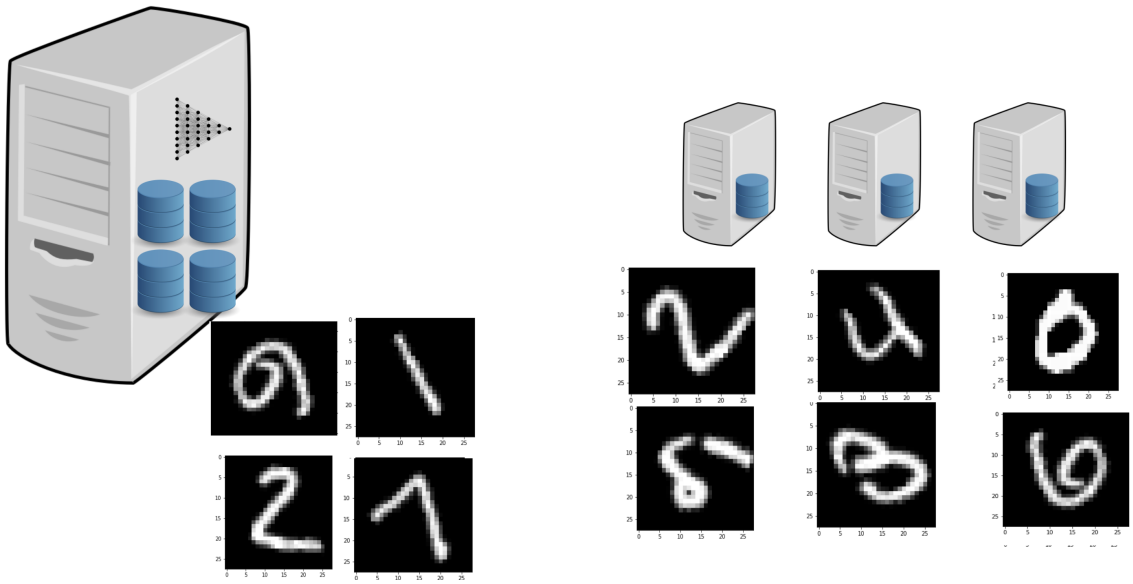

Figure 4.2 Federated version of MNIST with induced feature skew by rotating images from different classes and clients.

the algorithms have shown better accuracy than FedAVG in non-IID settings. Algorithms currently implemented in DomainBed were benchmarked in a federated setting, on CMNIST, and RotMNIST. The most important algorithm that we took through this test was

- Empirical Risk Minimization (ERM) minimizes the sum of errors across domains and examples.
- Meta-Learning for Domain Generalization (MLDG) leverages MAML to meta-learn how to generalize across domains.
- Invariant Risk Minimization (IRM) learns a feature representation such that the optimal linear classifier on top of that representation matches across domains. Apart from these, we also ran tests on IB-IRM, IB-ERM and VREx.

## 4.6 Experiments & Results

The aim of this project was to benchmark and compare the performance of existing OOD generalization algorithms while applied on the client side in a federated setting. To code base was highly inspired by the existing repository of DomainBed. The federated framework was borrowed from one of the first open-sourced implementations of Federated learning by fedavg-repo. Hence, our code base is at the core a combination of two highly popular and contributed to repositories in the area of machine learning. This choice was made keeping in mind the fact that these repositories are well maintained. Since the repository is not strange

for the community, it would make contributions and collaborations easier as well.

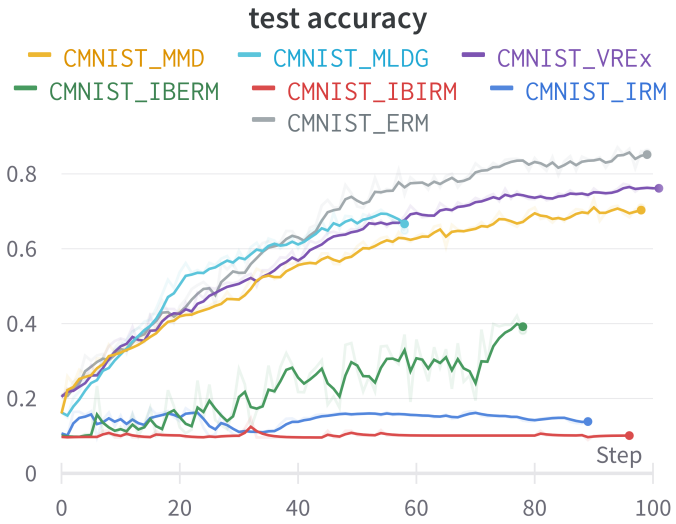


Figure 4.3 When applied on the client side of a federated setup, the existing OOD generalization algorithms show lower performance compared to ERM, and hence can be concluded to be inefficient in the setup.

All the experiments followed the same flow of control and repeated similar steps. The first step would be the initialization of the global model according to the dataset chosen. Since CMNIST is a 3-channel image dataset, the kernel of the MNIST-CNN model from DomainBed had to be altered to accept 3-channel input. Once the model is chosen, as many clones as the participating clients are created and distributed to clients. The second step was to translate the algorithm chosen through arguments, into an implementation of the *Algorithm* class. Hyper-parameters specific to the algorithm were also accepted through command line input. The logic was implemented in such a way that the choice of algorithms and dataset is validated before each run.

Due to the limitation in the available resources, all algorithms were tested in a setup with 10 participating clients and a central node. Since the number of clients was less by default, all clients were subject to training in every communication round; hence fraction of participating clients was always set to 1. Once the models have been distributed among the participating clients, training and weight updates would repeat as many rounds of local epochs that are being accepted as arguments for the run. Adam optimizer with a learning rate of 5e-5 was fixed for all the local epochs. In this step, the loss is calculated according to the OOD generalization algorithm being under consideration. The trained models are then aggregated using federated averaging (FedAVG). These aggregated models are then tested on a held-

out dataset, that is OOD to all the client datasets. The benchmark is dependent upon the performance of the global models trained with the specific algorithm chosen from DomainBed on this held-out central test dataset, after each communication round.

## 4.7 Conclusion

DOMAINBED [47] currently consists of datasets like RotatedMNIST [64], ColouredMNIST [51], DomainNet [58] and the rest, that are to be consumed by a single model in a centralized setting. The training environment consists of both invariant and spurious features; in the testing environment the invariant features are retained, whereas the spurious features are reversed.

FL involves multiple models being trained on various clients. Hence, the aforementioned mechanism for invariant and spurious features will not be sufficient. The first step of this project would be to define the invariant and spurious mechanisms for each dataset. Since the dataset is to be distributed across multiple clients, the spurious features would remain spurious, and the invariant features would remain invariant across clients and between train and test datasets. Furthermore, the dataset would be distributed among the participating clients in IID and non-IID settings using Dirichlet distribution as per the federated algorithm's requirement.

To represent the participating clients, we will need as many instances of the model being trained sequentially or in parallel. The aggregation step requires a server entity, which could perform the client model aggregations depending on the chosen federated optimization algorithm including but not limited to FedAvg [13], FedProx [65], SCAFFOLD [4], FedNova [66], FedGMA [67].

Since our concentration is on OoD generalization performance, the model evaluation would happen on the server in an OoD test environment. The algorithms currently listed in DOMAINBED [47] like IRM [51], ERM [68] MMD [69] and the rest, would be used to learn models on clients, with the aim of learning invariant features, which would then be aggregated at the server to enhance OoD generalization of the global model. This would inherently avoid the client models from learning spurious features and hence the global model would be devoid of them.

This project would consist of a wide range of experiments consisting of all combinations of OOD generalization techniques with federated optimization algorithms over all the listed datasets in both IID and non-IID data distributions. The same would be repeated across various federated hyper-parameters like the number of clients, local epochs, the fraction of

participating clients, and so on. We also aim to study the behavior of these settings under massive scaling of the network.

# CHAPTER 5 GENERAL DISCUSSION

Federated learning presents itself to be a method for data-private machine learning but it takes away the potential to produce high accuracy. The two potential factors affecting the reduced accuracy of federated learning could be

1. heterogeneity of data
2. inefficiency of the aggregator

## 5.1 FedSHIBU for personalized FL

This thesis dwells into the aforesaid reasons behind the lowered cap in accuracy and tries to mitigate it. Our initial experiment on training and aggregating models on class-specific data confirms that the heterogeneity of data hinders the models from reaching higher accuracy. FedSHIBU relies on Centered Kernel Alignment - an effective method to determine similarity in the behaviour of neural networks, to calculate data heterogeneity on clients. FedSHIBU uses this metric to effectively personalize the client models and outperforms the existing personalization techniques in federated learning.

### 5.1.1 Similarity-based aggregation

This thesis put forward the calculation of data similarity on clients and aggregating the models dependent on that metric as a method to utilize the diversity in clients and their data. For calculating similarity, a batch of client data of pre-determined size is sent through the client model to produce data embeddings - the output from the *extractor* of the model. These embeddings are then sent to a server which calculates the similarity of the received embeddings using CKA. The calculated similarity is then arranged into a matrix accessible for the server. The aggregation step makes use of this matrix to weigh the clients according to their similarity and assign personalized weights to each of them.

### 5.1.2 Scalability & potential risks

- **Is it scalable?** Scalability could become a concern in FedSHIBU since the algorithm uses representations from clients for the aggregation step. The representations can become a bottleneck for the network if the batch size becomes high. A small batch of

representations could result in lowered similarity matrix computation accuracy. This could also clog the network if the number of clients rises.

- **What are the security risks?** The presence of a similarity matrix to calculate the client weights is supposed to speed up the convergence for the clients. This increases the dependency of clients on each other. This results in a situation where the presence of an adversary in the network could be a riskier vulnerability than in a naive federated learning network. This is rooted in the fact that if an adversary generates representations similar to some clients the weights supplied by the adversary would be weighted higher, and hence results in increased chances for poisoning the network.
- **What are the privacy risks?** Sending the representations of data from clients to server(s) is a technique that doesn't fit with the core idea of federated learning. The potential threat that lies behind this is that the representations, even though are produced on clients' models with clients' data, could be used by an adversary to retrace the model, and even the data at times. For this reason, the presence of a strong encryption mechanism on top of the produced client representations could become mandatory for this idea to go into production. On the flip side, an encryption layer on the representations could result in erroneous similarity calculations and reduced accuracy.

## 5.2 Domain generalization benchmark

As a secondary direction, this thesis also presents the idea of building a benchmark for domain generalization of federated learning models. The benchmark is tailored around the idea of building a unified benchmark for comparing existing federated learning generalization algorithms. The project also proposes methods to introduce spurious features into existing public datasets and distribute them to clients in such a way that the clients hold personalized data. This skewed data is used to train local models on clients, and then the process proceeds to aggregate. During aggregation, the local models trained on client data would be aggregated and tested against a server-side dataset that contains a mix of invariant and spurious features compared to the clients. The results comparing some of the existing federated generalization techniques or algorithms are tabulated and presented.

# CHAPTER 6 CONCLUSION

## 6.1 Summary of Works

Federated learning has the potential to fix one of the most painful risks in machine learning - privacy. The in-built character of federated learning to not share data among the participating clients could be leveraged to ensure implicit security advantages over the existing distributed machine learning techniques. The same character of FL helps users carry out learning without clogging the network, especially in medical, automobile, and other big-data applications where data points are large in either size or quantity or both.

This thesis puts forward a method for better applying federated learning techniques in real-life scenarios where the distributed data is highly heterogeneous. The initial experiments help us validate the idea that clubbing models learned on similar classes of data would result in a higher personalization performance compared to a model which was learned irrespective of the source data it was trained on.

Further, we use a metric called CKA to compare the data representations from different clients to form a similarity matrix. CKA compares representations of data points and generates a decimal value that linearly represents the similarity between the data points being compared. The CKA values between data points from clients form a square matrix and the same is used to do a weighted aggregation of the models by the central server. This method has proven itself with higher accuracy than existing methods in extremely heterogeneous settings.

This thesis also presents a potential benchmarking system to measure the generalization abilities of federated learning algorithms. We put forward two synthetic datasets produced by introducing skews into existing public datasets, specific to the participating clients. These skews represent the spurious mechanisms that ideally shouldn't hinder the performance of the models. A model that claims high generalization should be able to intelligently learn only the invariant mechanisms and not overfit the spurious mechanisms. This benchmark is a pilot project that contributes towards building a unified method for measuring the generalization performance of federated learning.

## 6.2 Limitations

FedSHIBU brings about a potential solution to solve extreme data heterogeneity in a federated learning setting. But a considerable limitation of the project is scalability. The fact

that the similarity matrix grows to the $O(n^2)$ where $n$ is the number of participating clients would require higher computing on the server side to manage large networks. This could potentially stop the algorithm from being applied to cross-device networks where the number of participating clients in the network is of the order of 1000s and above.

### 6.3 Future Research

FedSHIBU being an extension of FedBABU improves the personalization performance of each participating client. This is made possible by consuming the differences in data representation across clients. It would be a good research direction to explore the potential privacy issues of sharing data representations from clients to the server. Even though clear data isn't shared, sharing the data representations may lead to the presence of security vulnerabilities. In this project, we have concentrated on homogenous models across clients. FedSHIBU already supports differences in models as long as it is the classifier that differs. An interesting direction for research would be to find out how to accommodate extractors that differ. Late joiners in the network, called stragglers [6], currently are not supported by FedSHIBU. In a real-life scenario, this is an important aspect to deal with. A potential solution for this would be to keep an averaged model using FedAVG [3] on the server at all times and distribute them to the late joiners to begin the training with. This could result in reduced performance in the first few rounds but eventually could improve as the training progresses. This is a quick-fix solution for the issue, but more thoughts could go in this avenue as well, to come up with a stronger and performance-oriented solution for the situation.

# REFERENCES


[1] P. Kairouz *et al.*, "Advances and open problems in federated learning," *CoRR*, vol. abs/1912.04977, 2019. [Online]. Available: http://arxiv.org/abs/1912.04977

[2] "How much data is created every day?" https://seedscientific.com/how-much-data-is-created-every-day/, written: October 28, 2021.

[3] H. B. McMahan *et al.*, "Communication-efficient learning of deep networks from decentralized data," 2016. [Online]. Available: https://arxiv.org/abs/1602.05629

[4] S. P. Karimireddy *et al.*, "Scaffold: Stochastic controlled averaging for federated learning," 2019. [Online]. Available: https://arxiv.org/abs/1910.06378

[5] T. Li *et al.*, "Federated optimization in heterogeneous networks," 2018. [Online]. Available: https://arxiv.org/abs/1812.06127

[6] J. Wang *et al.*, "Tackling the objective inconsistency problem in heterogeneous federated optimization," 2020. [Online]. Available: https://arxiv.org/abs/2007.07481

[7] I. Tenison, S. Francis, and I. Rish, "Gradient masked federated optimization," 2021. [Online]. Available: https://arxiv.org/abs/2104.10322

[8] F. Chen *et al.*, "Federated meta-learning with fast convergence and efficient communication," 2018. [Online]. Available: https://arxiv.org/abs/1802.07876

[9] C. T.Dinh *et al.*, "Fedu: A unified framework for federated multi-task learning with laplacian regularization," 02 2021.

[10] D. A. E. Acar *et al.*, "Federated learning based on dynamic regularization," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=B7v4QMR6Z9w

[11] G. Cheng, K. Chadha, and J. Duchi, "Federated asymptotics: a model to compare federated learning algorithms," 2021. [Online]. Available: https://arxiv.org/abs/2108.07313

[12] J. Oh, S. Kim, and S.-Y. Yun, "Fedbabu: Towards enhanced representation for federated image classification," 2021. [Online]. Available: https://arxiv.org/abs/2106.06042

[13] B. McMahan *et al.*, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[14] H. B. McMahan *et al.*, "Communication-efficient learning of deep networks from decentralized data," 2016. [Online]. Available: https://arxiv.org/abs/1602.05629

[15] B. Kang *et al.*, "Decoupling representation and classifier for long-tailed recognition," 2019. [Online]. Available: https://arxiv.org/abs/1910.09217

[16] A. Reisizadeh *et al.*, "Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization," 2019. [Online]. Available: https://arxiv.org/abs/1909.13014

[17] H. Yuan and T. Ma, "Federated accelerated stochastic gradient descent," 2020. [Online]. Available: https://arxiv.org/abs/2006.08950

[18] S. U. Stich, "Local sgd converges fast and communicates little," 2018. [Online]. Available: https://arxiv.org/abs/1805.09767

[19] J. Wang and G. Joshi, "Cooperative sgd: A unified framework for the design and analysis of communication-efficient sgd algorithms," 2018. [Online]. Available: https://arxiv.org/abs/1808.07576

[20] H. Yu, S. Yang, and S. Zhu, "Parallel restarted sgd with faster convergence and less communication: Demystifying why model averaging works for deep learning," 2018. [Online]. Available: https://arxiv.org/abs/1807.06629

[21] S. Wang *et al.*, "Adaptive federated learning in resource constrained edge computing systems," 2018. [Online]. Available: https://arxiv.org/abs/1804.05271

[22] T. Li *et al.*, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, may 2020. [Online]. Available: https://doi.org/10.1109%2Fmsp.2020.2975749

[23] T.-M. H. Hsu, H. Qi, and M. Brown, "Measuring the effects of non-identical data distribution for federated visual classification," 2019. [Online]. Available: https://arxiv.org/abs/1909.06335

[24] S. Reddi *et al.*, "Adaptive federated optimization," 2020. [Online]. Available: https://arxiv.org/abs/2003.00295

[25] H. Yuan *et al.*, "What do we mean by generalization in federated learning?" 2021. [Online]. Available: https://arxiv.org/abs/2110.14216

[26] S. Francis, I. Tenison, and I. Rish, "Towards causal federated learning for enhanced robustness and privacy," 2021. [Online]. Available: https://arxiv.org/abs/2104.06557

[27] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized federated learning: A meta-learning approach," 2020. [Online]. Available: https://arxiv.org/abs/2002.07948

[28] T. Li *et al.*, "Ditto: Fair and robust federated learning through personalization," 2020. [Online]. Available: https://arxiv.org/abs/2012.04221

[29] M. Duan *et al.*, "Astraea: Self-balancing federated learning for improving classification accuracy of mobile deep learning applications," in *2019 IEEE 37th International Conference on Computer Design (ICCD)*. IEEE, nov 2019. [Online]. Available: https://doi.org/10.1109%2Ficcd46524.2019.00038

[30] ——, "Astraea: Self-balancing federated learning for improving classification accuracy of mobile deep learning applications," in *2019 IEEE 37th International Conference on Computer Design (ICCD)*, 2019, pp. 246–254.

[31] D. A. E. Acar *et al.*, "Federated learning based on dynamic regularization," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=B7v4QMR6Z9w

[32] F. Yu *et al.*, "Heterogeneous federated learning," 2020. [Online]. Available: https://arxiv.org/abs/2008.06767

[33] C. Briggs, Z. Fan, and P. Andras, "Federated learning with hierarchical clustering of local updates to improve training on non-iid data," 2020. [Online]. Available: https://arxiv.org/abs/2004.11791

[34] V. Smith *et al.*, "Federated multi-task learning," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 4427–4437.

[35] Y. Chen *et al.*, "Fedhealth: A federated transfer learning framework for wearable healthcare," 2019. [Online]. Available: https://arxiv.org/abs/1907.09173

[36] C. T. Dinh, N. H. Tran, and T. D. Nguyen, "Personalized federated learning with moreau envelopes," 2020. [Online]. Available: https://arxiv.org/abs/2006.08848

[37] N. Kriegeskorte, M. Mur, and P. Bandettini, "Representational similarity analysis - connecting the branches of systems neuroscience," 2008. [Online]. Available: https://www.frontiersin.org/article/10.3389/neuro.06.004.2008

[38] I. R. O. Haroon Popal, Yin Wang, "A guide to representational similarity analysis for social neuroscience," 2020.

[39] H. Nili *et al.*, "A toolbox for representational similarity analysis," *PLOS Computational Biology*, vol. 10, 04 2014. [Online]. Available: https://doi.org/10.1371/journal.pcbi.1003553

[40] H. Dimsdale-Zucker and C. Ranganath, *Representational Similarity Analyses: A Practical Guide for Functional MRI Applications*, 01 2019, pp. 509–525.

[41] P. B. Nikolaus Kriegeskorte, Marieke Mur, "Representational similarity analysis – connecting the branches of systems neuroscience," 2008.

[42] K. Dwivedi and G. Roig, "Representation similarity analysis for efficient task taxonomy & transfer learning," 2019. [Online]. Available: https://arxiv.org/abs/1904.11740

[43] A. S. Morcos, M. Raghu, and S. Bengio, "Insights on representational similarity in neural networks with canonical correlation," 2018. [Online]. Available: https://arxiv.org/abs/1806.05759

[44] S. Kornblith *et al.*, "Similarity of neural network representations revisited," 2019. [Online]. Available: https://arxiv.org/abs/1905.00414

[45] M. Dujmović *et al.*, "The pitfalls of measuring representational similarity using representational similarity analysis," *bioRxiv*, 2022.

[46] R. Martin Cichy *et al.*, "Dynamics of scene representations in the human brain revealed by magnetoencephalography and deep neural networks," *NeuroImage*, vol. 153, pp. 346–358, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1053811916300076

[47] I. Gulrajani and D. Lopez-Paz, "In search of lost domain generalization," *arXiv preprint arXiv:2007.01434*, 2020.

[48] X. Qiu *et al.*, "A first look into the carbon footprint of federated learning," 02 2021.

[49] N. Truong *et al.*, "Privacy preservation in federated learning: An insightful survey from the gdpr perspective," *Computers Security*, vol. 110, p. 102402, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404821002261

[50] M. Chen *et al.*, "Communication-efficient federated learning," *Proceedings of the National Academy of Sciences*, vol. 118, no. 17, p. e2024789118, 2021. [Online]. Available: https://www.pnas.org/doi/abs/10.1073/pnas.2024789118

[51] M. Arjovsky *et al.*, "Invariant risk minimization."

[52] D. Krueger *et al.*, "Out-of-Distribution Generalization via Risk Extrapolation (REx)," *arXiv e-prints*, p. arXiv:2003.00688, Mar. 2020.

[53] G. Parascandolo *et al.*, "Learning explanations that are hard to vary," *arXiv e-prints*, p. arXiv:2009.00329, Sep. 2020.

[54] S. Shahtalebi *et al.*, "Sand-mask: An enhanced gradient masking strategy for the discovery of invariances in domain generalization," 2021. [Online]. Available: https://arxiv.org/abs/2106.02266

[55] K. Ahuja *et al.*, "Invariance principle meets information bottleneck for out-of-distribution generalization," 2021. [Online]. Available: https://arxiv.org/abs/2106.06607

[56] J. Deng *et al.*, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[57] K. Zhou *et al.*, "Deep domain-adversarial image generation for domain generalisation," 2020. [Online]. Available: https://arxiv.org/abs/2003.06054

[58] B. Neyshabur, H. Sedghi, and C. Zhang, "What is being transferred in transfer learning?" 2020. [Online]. Available: https://arxiv.org/abs/2008.11687

[59] M. M. Rahman *et al.*, "Multi-component image translation for deep domain generalization," 2018. [Online]. Available: https://arxiv.org/abs/1812.08974

[60] S. Sagawa *et al.*, "Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization," 2019. [Online]. Available: https://arxiv.org/abs/1911.08731

[61] Y. He, Z. Shen, and P. Cui, "Towards non-iid image classification: A dataset and baselines," *Pattern Recognition*, vol. 110, p. 107383, 2021.

[62] I. Gulrajani and D. Lopez-Paz, "In search of lost domain generalization," 2020. [Online]. Available: https://arxiv.org/abs/2007.01434

[63] J.-C. Gagnon-Audet *et al.*, "Woods: Benchmarks for out-of-distribution generalization in time series tasks," 2022. [Online]. Available: https://arxiv.org/abs/2203.09978

[64] M. Ghifary *et al.*, "Domain generalization for object recognition with multi-task autoencoders," *CoRR*, vol. abs/1508.07680, 2015. [Online]. Available: http://arxiv.org/abs/1508.07680

[65] A. K. Sahu *et al.*, "On the convergence of federated optimization in heterogeneous networks," *CoRR*, vol. abs/1812.06127, 2018. [Online]. Available: http://arxiv.org/abs/1812.06127

[66] J. Wang *et al.*, "Tackling the objective inconsistency problem in heterogeneous federated optimization," *CoRR*, vol. abs/2007.07481, 2020. [Online]. Available: https://arxiv.org/abs/2007.07481

[67] I. Tenison *et al.*, "Gradient masked averaging for federated learning," *CoRR*, vol. abs/2201.11986, 2022. [Online]. Available: https://arxiv.org/abs/2201.11986

[68] V. N. Vapnik, *Statistical Learning Theory*. Wiley, 1998.

[69] H. Li *et al.*, "Domain generalization with adversarial feature learning," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5400–5409.