PROTOTYPE LEARNING BASED FEDERATED LEARNING

by

Alperen Yıldırım

B.S., Industrial Engineering, Boğaziçi University, 2019

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering
Boğaziçi University

2025

ACKNOWLEDGEMENTS

First and foremost, I would like to express my deepest gratitude to my advisor, Assist. Prof. İnci Meliha Baytaş, for her unwavering support, invaluable guidance, and encouragement throughout my journey. Her dedication to my success has been an immense source of inspiration, and her expertise has profoundly shaped the direction of my research.

I would like to extend sincere gratitude to Assoc. Prof. Atay Özgövde and Assoc. Prof. Günce Orman for accepting to serve as my thesis committee member and share their valuable feedback to improve my thesis.

I would also like to extend my heartfelt thanks to my beloved wife, Şevval Serdaroğlu Yıldırım, and my son, Atlas Yıldırım, for their boundless love, patience, and endless belief in me. Their presence has been my greatest source of strength and motivation throughout this challenging journey. I am equally grateful to all my family members for their endless encouragement, support, and faith in me.

Finally, I extend my sincere appreciation to everyone who has supported me during this journey, whether through advice, collaboration, or moral support. Each contribution, no matter how small, has played a significant role in the completion of this thesis.

ABSTRACT

PROTOTYPE LEARNING BASED FEDERATED LEARNING

Heterogeneous data pose various challenges in personalized federated learning frameworks, where customized local solutions are required for each client. In such scenarios, customizing a global feature representation by local classifiers in each client is one of the approaches to tackling the challenge. Nevertheless, incorporating the global information often obtained by some aggregation strategy over several clients might restrict the generalization capability of the client models when there are substantial drifts between the local models.

This thesis proposes an attention strategy for aligning global and local prototypes in the federated prototype learning setup. The proposed framework aggregates the weighted prototypes where the attention weights are computed client-specific. Thus, prototypes for each client are aggregated based on prototype similarities with the other clients.

In addition, the local model training is regularized with the discrepancy between the global and local prototypes to align local and global information without sacrificing personalization. The proposed framework is evaluated with well-known benchmark datasets in various heterogeneous data scenarios. The experiments report significant performance gains compared to the state-of-the-art.

ÖZET

PROTOTİP ÖĞRENME TEMELLİ FEDERE ÖĞRENME

Heterojen veriler, federated learning (dağıtık öğrenme) konusunda çeşitli zorluklar oluşturur ve bu durumda her bir istemci için özelleştirilmiş yerel çözümler gereklidir. Böyle senaryolarda, her istemcideki yerel sınıflandırıcılar tarafından özelleştirilmiş bir küresel özellik temsili oluşturmak, bu zorluğun üstesinden gelmek için uygulanabilir yaklaşımlardan biridir. Bununla birlikte, birkaç istemci üzerinde birleştirme stratejisi ile elde edilen küresel bilgiyi içermek, yerel modeller arasında önemli sapmalar olduğunda istemci modellerinin genelleştirme yeteneğini sınırlayabilir.

Bu çalışma, federated prototype learning (dağıtık prototip öğrenme) yapısında küresel ve yerel prototipleri hizalamak için bir dikkat stratejisi önermektedir. Önerilen çerçeve, dikkat ağırlıklarının istemciye özgü olarak hesaplandığı prototipleri birleştirir. Bu nedenle, önerilen yaklaşım, yerel özellikleri kapsayan küresel bir temsil oluşturmayı hedefler ve aynı zamanda potansiyel veri gizliliği ihlallerine karşı artan bir savunmasızlık oluşturmaz.

Buna ek olarak, yerel model eğitimi, istemciye özgü bir şekilde çıkarılan küresel bilgiden faydalanabilir.Önerilen çerçeve, çeşitli heterojen veri senaryolarında iyi bilinen kıyaslama veri setleriyle değerlendirilmiştir. Yapılan kapsamlı deneyler, güncel yöntemlere kıyasla önemli performans artışları rapor etmektedir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

$\frac{1}{N_i^c}$	Denotes the training set size
$\mathbf{b}$	Bias vector
D_i^c	Training set of the category c of client i
$\mathbb{E}_{(\mathbf{x},y)\sim p_{XY}^{(i)}}$	Expected loss for client i
$f_i(\mathbf{x})$	Feature vector of an input $\mathbf{x}$
$\mathbf{k}_j^c$	Key
$\ell(\mathbf{w}_i; \mathbf{x}, y)$	A supervised loss function
$\mathbf{p}_i$	Local prototype
$\mathbf{p}_i^c$	Prototype for class c from client i
$\mathbf{p}_i^{\text{global}}$	Global prototype aggregated
$\mathbf{p}_{\text{agg}i}^c$	The aggregated prototype for client i and class c
$\mathbf{q}_i^c$	Query
$\mathbf{v}_j^c$	Value
$\mathbf{W}_K$	A learnable weight of key
$\mathbf{W}_Q$	A learnable weight of query
$\mathbf{W}_V$	A learnable weight of value
$\hat{y}_{i,k}$	Predicted probability
$y_{i,k}$	Ground truth label
α_{ij}^c	Attention weights
$\rho_{XY}^{(i)}$	Local data distribution of client i

LIST OF ACRONYMS/ABBREVIATIONS

FedASA	Federated Learning Aggregation with Self-Attention
FedASALD	Federated Learning Aggregation with Self-Attention Large Layers
FedASALW	Federated Learning Aggregation with Self-Attention Label Weights
FedASAWA	Federated Learning Aggregation with Self-Attention Weight Aggregation
FedAvg	Federated Averaging
FedBABU	Federated Bias Adjustment with Representation Learning
FedDyn	Federated Learning with Dynamic Regularization
FedKTL	Federated Knowledge Transfer Learning
FedPAC	Personalized Federated Learning with Feature Alignment and Classifier Collaboration
FedProto	Federated Prototype Learning
FedProx	Federated Proximal Optimization
FedROD	Federated Regularized Optimization for Decentralized Data
FedTGP	Federated Trainable Global Prototypes
FedTP	Federated Transformer Personalization
FL	Federated Learning
non-IID	Non-Independent and Identically Distributed
PFL	Personalized Federated Learning
SVHN	Street View House Numbers

1. INTRODUCTION

In the last three decades, we have witnessed rapid advancements in machine learning and deep learning fields. These techniques have been successfully applied to various tasks of varying levels of complexity, such as face and object recognition, and generative artificial intelligence applications. The state-of-the-art performances are often obtained in a centralized setting where a vast amount of data should be centralized and available in a local machine during model training. However, in real-world problems, data has a distributed nature. In various applications, such as edge intelligence and the Internet of Things, data is collected locally in a heterogeneous fashion, where the hardware specifications of local devices, speed of data collection, and local data distributions vary.

Centralizing local data may not be possible in a distributed data setting due to several factors. Privacy concerns and bandwidth constraints are some of the key limitations. Sensitive data, including medical records, financial transactions, and personal user information, are usually protected by strict legal frameworks and regulations limiting local data sharing, storage, and utilization. Furthermore, self-driving cars, IoT, and mobile applications continuously generate vast data. Transferring this data to a central server might cause significant bandwidth costs, latency, and data loss during transmission. Machine learning and deep learning model training require a centralized dataset of independent and identically distributed (IID) samples. However, distributed data for real-world applications often violates the IID assumption. Federated learning (FL) aims to address the distributed data challenge through an optimization framework that enables collaborations between separate models trained with local datasets. The federated learning frameworks can be divided into several categories based on data partitioning and networking structure [1]. Horizontal FL, vertical FL, and federated transfer learning are three fundamental types of FL frameworks based on how data is distributed.

In horizontal FL, client datasets have the same feature space but different instances, e.g., patient datasets in different hospitals where the same features are recorded. On the other hand, the vertical FL is concerned with clients that share the same instances but different features. Finally, the distributed data comprises different features and distributions in federated transfer learning. In this scenario, transfer learning can be utilized to enable collaboration between clients [2].

Depending on their networking structure, FL frameworks can be centralized and decentralized [1]. In the centralized FL setting, the central server coordinates the model aggregation and global updates. It forecasts the global model to the clients, which works parallel to improve the global model. On the other hand, decentralized FL does not require a central server to guide the optimization framework [1]. Each client trains their local models and shares them with other clients. Thus, clients are responsible for model aggregation and update through peer-to-peer communication.

The data distribution setup considered in this thesis is related more to federated transfer learning than horizontal and vertical FL. Our experiments consider different levels of distributional shift between client datasets. Therefore, the proposed framework does not require an overlap between the instances of the local datasets. The feature spaces do not need to be the same, either. Regarding networking structure, the proposed FL framework is an instance of a centralized FL where a central server is required to aggregate local information. In addition to the data distribution and networking structure, a classification of FL frameworks, namely canonical and personalized FL, has emerged based on the generalizability of the aggregated model [3].

Canonical FL refers to the traditional FL setup where clients work collaboratively to improve a global model. For instance, the well-known FedAvg framework proposed by McMahan et al. aggregates client models to obtain a single global model without sharing local datasets with the server [4]. On the other hand, the goal is not always to obtain a generalized global model for inter-clients.

When the aim is to improve local performance, a personalized FL (PFL) approach is more effective than a canonical framework. The PFL frameworks aim to improve the client's performance by benefiting from the shared knowledge between clients while preserving client-specific properties of the local models [3].

Despite its potential, FL encounters challenges, such as handling the non-IID data distributions and achieving a perfect equilibrium between global generalization and local customization. Severe distributional shifts between the clients might severely degrade the personalized performances. Therefore, in contrast to the canonical FL, which mainly focuses on global generalization, PFL attempts to balance the global model and local personalization. Various PFL frameworks have been investigated in the literature. For instance, clustering-based approaches group clients with similar data distributions [5], multi-task learning approaches consider each client's model as a task [6], transfer learning-based frameworks utilize pre-trained models [7], and regularization techniques [3] modify the global model based on client-specific data.

One of the common approaches in PFL is splitting the client's architecture into a body and a head, focusing on global and local information, respectively. This approach allows the body, the collective entity, to identify and understand broad trends across all clients. Whereas the head, the personalized component of the system, fine-tunes the model to accommodate the unique characteristics of each client. Thus, a global model can be used in various scenarios while enhancing the model's suitability and efficiency when applied to individual client data.

This thesis focuses on the PFL setting under various non-IID client data distributions. The proposed PFL framework, FedASA, focuses on aligning the global feature representation with client-specific representation by learning client-specific prototypes and aggregating them on the server. Client-specific prototypes are globally updated using self-attention in a client-aware fashion where each client's prototype contributes to other client's prototypes differently. Thus, the proposed framework aims to alleviate the adverse effects of non-IID data on the personalization of the client models while

sharing knowledge between different clients. The primary emphasis of the thesis is on the challenges presented by the existence of heterogeneous client data. The main contributions of our work are as follows:

- A new approach for calculating global prototypes by leveraging self-attention is proposed. This approach utilizes the self-attention mechanism [8] to compute a weighted combination of client prototypes.
- Different sets of attention weights are computed for each client to obtain client-specific global prototypes. Thus, the proposed framework aims to align the global updates with the client's needs.
- The proposed approach enables clients to carry out additional localized updates while ensuring coherence with the global model. This minimizes the disparity between local and global models, resulting in enhanced representation learning in a manner that conserves communication resources.
- Experiments on well-known datasets under various levels of non-IID conditions are conducted. The FedASA consistently outperforms state-of-the-art PFL models or performs close to them.
- Unlike most baselines, the FedASA provides an opportunity to investigate which clients have a more significant impact on the global prototype aggregation for each client via the attention weights. Thus, the proposed model is inherently interpretable in terms of client contributions.

2. RELATED WORK

The heterogeneous characteristics of client data can result in significant variations in the local optimal solutions obtained by each client from their datasets, potentially causing significant variances from the global optimal solution. This divergence occurs due to the non-IID characteristics of client data, resulting in distinct data distributions for each client, leading to localized model updates that may not consistently match with the global optima. This results in a difference between local and global parameters called client drift. Client drift has affected the result of FedAvg [4] drastically [9].

FedAvg [4] has become a traditional FL framework and a baseline. Various improvements have been proposed over FedAvg. These improvements can be applied to either the global aggregation step or the local training step. For example, FedProx [10] and FedDyn [11] use regularization to reduce the drift of local models from the global model, while SCAFFOLD [12] uses control variates for the same purpose. Another approach is to apply objective functions from class-imbalanced learning, which are designed to be robust to changes in class distribution. This approach is similar to the one used by Hsu et al. [13], where a traditional class-imbalanced treatment called re-weighting was employed.

Specific methodologies aim to eliminate bias from clients or servers by utilizing gradients. FedProx [10] normalizes client gradients prior to aggregation by leveraging the disparity in parameters between the server and client. Drawing inspiration from momentum SGD, Yu et al. employ a local buffer to accumulate gradients from previous iterations at each client, subsequently communicating this momentum buffer alongside local parameters to the server, thereby effectively doubling communication bandwidth consumption [14]. Alternatively, some approaches implement server momentum, which mitigates the increase in communication costs [13,15]. Knowledge distillation is used in some methods of FL for heterogeneous data by ensemble client models to train a server model to enhance model performance. This approach is more effective than simple

parameter averaging in reducing the bias of the local gradients [16–18]. Additionally, SCAFFOLD [12] employs control variates on the server and clients to mitigate variation in local updates. Alongside the model parameters, the control variates are acquired and transmitted between the server and the clients, consuming additional bandwidth.

PFL frameworks aim to mitigate the data constraints inherent in conventional FL, wherein a singular global model is developed and utilized across all clients. Many PFL approaches have been proposed in the literature, including clustering, multi-task learning, transfer learning, regularized loss functions, and meta-learning. These strategies often focus on improving the generalization of local models by exchanging information among associated clients. In particular, Briggs et al. and Mansour et al. used clustering to categorize similar clients and developed distinct models for each cluster without inter-cluster federation [5, 6].

Multi-task learning-based PFL studies involve learning models for associated clients without dividing them into clusters. Thus, an exchange of representations among associated clients enhancing the generalization of each model can be attained. Smith et al. [19] and Dinh et al. [20] showed that multi-task learning is effective for individualized federated learning. Transfer learning is also suggested for PFL to facilitate the exchange of knowledge among clients. Yang et al. [21] and Chen et al. [7] employed transfer learning to improve local models by sharing data among selected clients. Furthermore, Dinh et al. [22] and Li et al. [23] utilized regularization to reduce the risk of local models overfitting to their client datasets. Both models use a loss for the difference between the global and local models to ensure personalized models remain close to the global ones while trying to learn local properties.

FedPer [24] and FedPav [25] are two methods proposed for PFL that entail collaboratively learning the complete network during local updates while exclusively aggregating the lower layers. This method resembles FedRep [26], which similarly learns the complete network sequentially during local updates but exclusively aggregates the body network. During the local update phase, each client initially receives the head

from the aggregated representation and then updates the body using its head in a single epoch. Thus, customized models tailored to each client can be obtained.

There are also other methods utilizing feature representations to provide personalized federated learning. In FedProto [27], the clients and server communicate abstract class prototypes rather than gradients, which are used to regularize the training of the local models. The training aims to minimize the classification error on the local data while keeping the local prototypes close to the global ones. FED-ROD [28] employs a class-balanced loss for training the local model, improving the global model's robustness over non-IID data. A personalized predictor is also introduced and trained using the client's empirical risk to ensure optimal performance in personalized federated learning tasks. The personalized model utilizes a hypernetwork to parameterize the class distribution of the client, allowing the model to adjust to new clients without requiring data and offering an improved initialization for fine-tuning on fresh client data. In FedBABU, the local update phase emphasizes modifying the model's body using the client's data, while the model's head remains static [3]. The fixed head functions as a framework of criteria for each class, while the model's body is trained to acquire representations.

The FEDPAC method, presented by Xu et al., represents an important step forward in customized FL by integrating feature alignment and classifier collaboration [29]. This architecture improves the traditional FL configuration by utilizing a shared feature representation and customizing classifier heads for specific clients, unlike the typical approach where local updates may not efficiently leverage global information. The FEDPAC framework strategically utilizes global semantic knowledge to synchronize local characteristics to enhance the consistency between the global model and personalization. This method also aims to enhance classifier aggregation using a quadratic programming formulation to minimize predicted testing loss. In FedTGP, Zhang et al. aim to address both sides of data and model heterogeneity in federated learning. The method improves alignment between global representations and client-specific data distributions by implementing trainable global prototypes and utilizing an adaptive-

margin-enhanced contrastive learning mechanism [30]. This method highlights the importance of shared prototypes in enhancing customization and generalization among diverse clients, providing insights into advanced prototype-based solutions in federated learning. On the other hand, FedKTL presents an innovative method to enhance communication efficiency in FL by utilizing a pre-trained generator at the server. This approach mitigates the constraints caused by diverse data distributions by efficiently delivering knowledge from the server to clients with minimum upload expenses [31]. Thus, FedKTL enhances communication efficiency and information transmission, offering an additional view to current approaches that emphasize either model aggregation or local optimization in federated environments. Its focus on minimizing communication overhead while preserving performance despite heterogeneity makes it a significant contribution to the literature.

The FedTP utilizes transformer-based architectures to improve customization in federated learning. The method employs the self-attention mechanism to adapt client models to their local data distributions while preserving a common global representation [32]. FedTP tackles the issues of non-IID data distributions and client heterogeneity by training a hypernetwork that outputs client-specific projection layers to obtain key, query, and value embeddings. Thus, the FedTP framework aims to learn how to personalize client self-attention layers from a shared global layer. This thesis also focuses on a self-attention approach. However, unlike FedTP, the proposed approach learns how to personalize the prototype aggregation step without sending any global weights to the clients.

Compared to the related studies summarized in this section, this thesis focuses more on personalizing global updates rather than leaving the personalization to clients only. The proposed global prototype generation approach aims to improve its adaptability and effectiveness across different degrees of data heterogeneity and client availability, reflecting the diverse real-world applications of FL. The proposed client framework also follows regularized training procedures.

3. METHODS

The proposed prototype learning-based FL framework, FedASA, aims to alleviate the challenges encountered in PFL due to data heterogeneity by customizing global prototype updates based on self-attention. The FedASA, given in Figure 3.1, learns to globally aggregate client-specific prototypes by learning separate self-attention scores for each client and category in a supervised learning setting. Then, the updated prototypes in a global but client-aware fashion are used by the clients to improve the training of local models, ensuring better adaptation to non-IID data distributions while maintaining global consistency. Figure 3.1 outlines the global and local optimization frameworks. The following sections present the problem setup and client-side and server-side workflows.

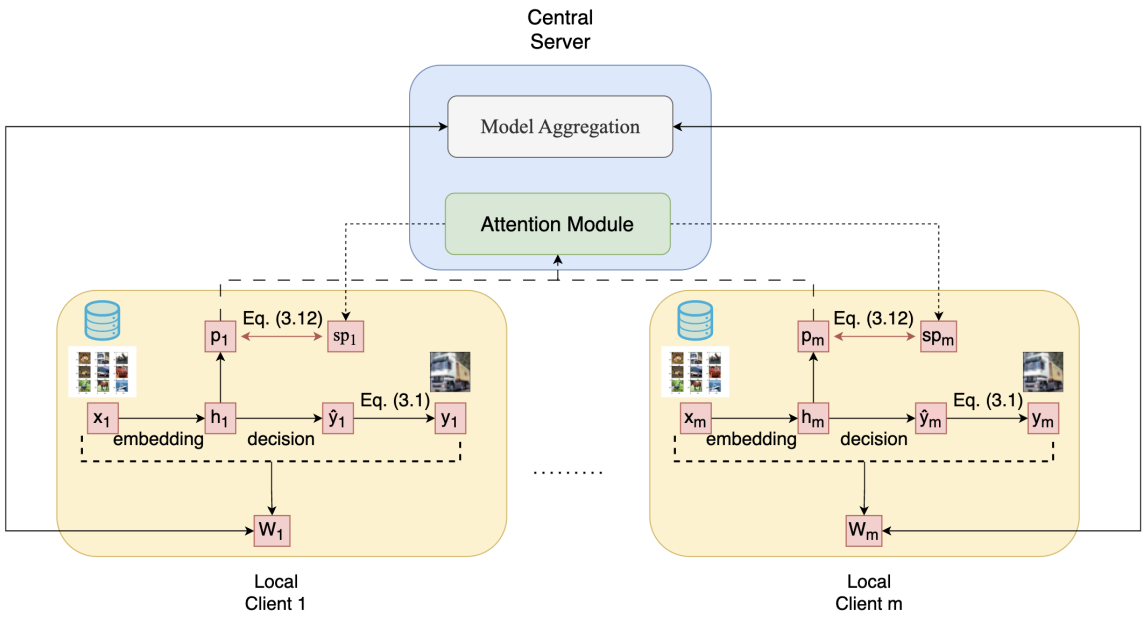

Figure 3.1. An overview of proposed FedASA framework. Each client calculates its prototypes, p, and then sends them to the central server. The global attention module updates the prototypes for each client. The global prototype updates, sp, are then sent back to the clients where the global and local prototypes are aligned during client model training.

3.1. Problem Setup

The FL setting considered in this thesis is comprised of a central server and m clients, each possessing its own local dataset. The central server coordinates the training process without directly accessing the clients' raw data, thereby maintaining data privacy. Each client i is characterized by a local data distribution $\rho_{XY}^{(i)}$, defined over the input space X and label space Y. The objective is to train a set of personalized client models $\mathbf{W} = \{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_m\}$. Each model is trained using a supervised loss function $\ell(\mathbf{w}_i; \mathbf{x}, y)$, where data point $(\mathbf{x}, y)$ is sampled from $\rho_{XY}^{(i)}$.

The primary goal is to minimize the aggregated loss

$$\min_{\mathbf{W}} \left\{ F(\mathbf{W}) := \frac{1}{m} \sum_{i=1}^{m} \mathbb{E}_{(\mathbf{x},y)\sim p_{XY}^{(i)}} \left[\ell(\mathbf{w}_i; \mathbf{x}, y) \right] \right\} \tag{3.1}$$

across all clients. Client weights might correspond to a neural network architecture designed to solve the personalized task. This thesis considers image classification as the client task, and the local datasets are image datasets. Each client model is locally trained while leveraging the collective knowledge shared among the clients.

3.2. Client-Side Workflow

Each client trains a convolutional neural network (CNN) with their local datasets for the image classification task. For this purpose, two convolutional layers succeeded by fully connected layers given in Table 3.1 are considered. The client architecture, except for the last fully connected layer, is considered the base model. The last fully connected layer is the classifier model head. Base and head model parameters are initialized in the server and sent to clients at the beginning of the federated learning rounds.

The training procedure for the client is divided into two steps inspired by FedRep [26]. First, the model head is trained with the local dataset to minimize the loss

given below while the base model is frozen for a fixed number of epochs:

$$\mathcal{L}_{\text{head}} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{C} y_{i,k}\log(\hat{y}_{i,k})\,, \tag{3.2}$$

where N is the total number of samples on a client, C is the number of categories, $y_{i,k}$ is the ground truth label, and $\hat{y}_{i,k}$ is the predicted probability. This step allows the client to personalize the classifier to its local data distribution without altering the shared global feature representation. Once the head is trained, the second step begins,

Table 3.1. Network Architecture of the Client Model. The output size of the last fully connected layer, C, varies depending on the dataset.

Layer	**#Channels**	**Filter Size**	**Activation**	**Output Size**	**#Parameters**
Input Layer	-	-	-	$28 \times 28 \times 1$	-
Convolutional	32	5×5	ReLU	$24 \times 24 \times 32$	832
Max Pooling	-	2×2	-	$12 \times 12 \times 32$	-
Convolutional	64	5×5	ReLU	$8 \times 8 \times 64$	51,264
Max Pooling	-	2×2	-	$4 \times 4 \times 64$	-
Flatten	-	-	-	1×1024	-
Fully Connected	-	-	ReLU	1×512	524,800
Fully Connected	-	-	Softmax	$1 \times C$	C

where the base model is trained with the local dataset in order to minimize the loss function given below:

$$\mathcal{L}_{\text{base}} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{C} y_{i,k}\log(\hat{y}_{i,k}) + \lambda\mathcal{L}_{\text{proto}}\,, \tag{3.3}$$

where the cross-entropy loss is used, λ is the regularization parameter, and $\mathcal{L}_{\text{proto}}$ is the prototype regularization term. At this process model head is frozen.

The prototype regularization term is inspired by FedProto [27]. It is the mean squared error objective used to align the global prototypes sent by the server and the local feature representations as given below:

$$\mathcal{L}_{\text{proto}} = \frac{1}{N}\sum_{i=1}^{N} \|\mathbf{p}_i - \mathbf{p}_i^{\text{global}}\|_2^2\,, \tag{3.4}$$

where $\mathbf{p}_i$ is the local prototype and $\mathbf{p}_i^{\text{global}}$ is the global prototype aggregated in a client-specific way. For each category, a global prototype is obtained. Therefore, the global prototype of the category that matches the label of the local prototype is used in Equation 3.4. The regularization synchronizes local representations with the global ones, alleviating the impact of data heterogeneity.

Each client c_i first trains a local model using its private dataset, resulting in client-specific prototypes for each class. For each client, a class prototype per category is obtained at the end of the local model training phase. A class prototype is computed as the mean of the embedding vectors of the training instances of the corresponding class [27]. Let $f_i(\mathbf{x}) \in \mathbb{R}^d$ denote the d-dimensional feature vector of an input $\mathbf{x}$ obtained from the i-th client's base model, $f_i(\cdot)$. The class prototype is computed for each client i as follows in order to send to server:

$$\mathbf{p}_i^c = \frac{1}{N_i^c} \sum_{\mathbf{x}_j \in D_i^c} \mathbf{f}_i(\mathbf{x}_j) \,, \tag{3.5}$$

where $\frac{1}{N_i^c}$ and D_i^c denote the training set size and the training set of the category c of client i, respectively. These prototypes are then communicated to the server along with the client model parameters for global prototype and model weight aggregation.

3.3. Server-Side Workflow

The server orchestrates the training rounds and chooses a subset of clients to engage in each round. It gathers the prototypes and local models from the clients and propagates the updated prototypes and the aggregated global model back to the clients. Prototypes are aggregated by a self-attention mechanism proposed in this thesis that learns the contributions of client prototypes to each client's prototype updates.

This weighted aggregation produces a refined collection of global prototypes, which are sent to clients for the next rounds of client training. The model aggregation step follows averaging the client models as in FedAvg [4]. The training procedure summarized in Figure 3.2 alternates between client and server for a predetermined number of rounds. The server begins by initializing the global model and setting hy-

perparameters such as the learning rate and regularization coefficients. During each communication round, the server interacts with the clients by sending the global model and receiving client-specific prototypes $\mathbf{P}_i = \{\mathbf{p}_i^c \mid c = 1, \ldots, C\}$, where $\mathbf{p}_i^c$ represents the prototype for class c from client i. These prototypes are aggregated to create a globally representative set of prototypes while retaining the specificity of each client's data distribution.

A key feature of the server is the aggregation of client prototypes using a self-attention mechanism, which assigns client and category-specific weights to each prototype based on its relevance to other clients. The attention scores are computed as the dot product between keys and queries of the prototypes. The use of attention mechanisms allows the framework to dynamically adjust the importance of different clients' contributions based on the context and the relevance of their data. In the first step of the proposed prototype aggregation approach, prototype embeddings are mapped to key, query, and value vectors. Firstly, the query vector is calculated as follows:

$$\mathbf{q}_i^c = \mathbf{W}_Q \mathbf{p}_i^c + \mathbf{b}_Q \tag{3.6}$$

Then the key vector is calculated using prototypes

$$\mathbf{k}_j^c = \mathbf{W}_K \mathbf{p}_j^c + \mathbf{b}_K \tag{3.7}$$

which then will be used in score calculation. Lastly, the value vector

$$\mathbf{v}_j^c = \mathbf{W}_V \mathbf{p}_j^c + \mathbf{b}_V \tag{3.8}$$

is calculated in order to use in aggregated prototype calculation. The parameters are $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V$ are learnable weight matrices, and $\mathbf{b}_Q, \mathbf{b}_K, \mathbf{b}_V$ are bias vectors.

Attention scores are computed as the dot product of the query of the client of interest i and the key vectors of the remaining available clients j:

$$\text{Score}(i, j) = \frac{\mathbf{q}_i^c \cdot \mathbf{k}_j^c}{\sqrt{d_k}}, \tag{3.9}$$

where d_k is the dimensionality of the key vectors. Next, scores are normalized using the softmax function to compute the client-specific attention weights for each category:

$$\alpha_{ij}^c = \text{softmax}(\text{Score}(i, j)) \tag{3.10}$$

Finally, using the attention weights, the global prototype for class c and client i, $\mathbf{p}^c_{\text{agg}i}$, is computed by using the below formula:

$$\mathbf{p}^c_{\text{agg}i} = \sum_{j=1}^{m} \alpha^c_{ij} \mathbf{v}^c_j \,, \tag{3.11}$$

where m is the number of clients. The attention-based aggregation mechanism aims for a personalized global aggregation of the client prototypes by allowing cross-client attention where the query is different for each client to address the data heterogeneity. Loss for the attention mechanism is as follows:

$$\mathcal{L} = \frac{1}{m} \sum_{i=1}^{m} \sum_{c} \left\| \mathbf{p}^c_{\text{agg}i} - \mathbf{p}^c_i \right\|^2 \,, \tag{3.12}$$

where $\mathbf{p}^c_{\text{agg}i}$ is the aggregated prototype for client i and class c, computed as a weighted sum of prototypes from all clients. The weights α^c_{ij} represent the similarity between the prototypes of client i and client j for class c. Real prototype for client i and class c is represented by $\mathbf{p}^c_i$. Server trains this self attention mechanism through the all training process. Equation 3.12 is used to learn weights. The attention mechanism starts its training at the server from scratch at the first communication round. After the first one, we continue to train attention-mechanism through all communication rounds. At each round the model is trained for 50 epochs. After each training the final aggregated prototypes are calculated as result of equation 3.11. The aggregated prototypes and the updated global model are sent back to the clients to guide their next round of local training. This iterative process given in Figure 3.2 continues for a fixed number of communication rounds T or until convergence is achieved. The server continuously integrates client contributions through prototype and model aggregation and controls for the client accuracies. This approach balances local personalization and global consistency.

Algorithm 1 `FedAsa`

Server executes:

Initialize global model weights, $\{\mathbf{W}^i_{\text{head}}, \mathbf{W}^i_{\text{base}}\}_{i=1}^m$ and global prototype set $\bar{P} = \{\mathbf{P}_1, \cdots, \mathbf{P}_m\}$ for m clients and for C classes, where $\mathbf{P}_i = \{\mathbf{p}_1, \cdots, \mathbf{p}_C\}$.
for each round $T = 1, 2, ...$ **do**
    for each client i **in parallel do**
        $\mathbf{P}_i, \mathbf{W}^i_{\text{head}}, \mathbf{W}^i_{\text{base}} \leftarrow$ LocalUpdate $(i, \mathbf{P}_i, \mathbf{W}^i_{\text{head}}, \mathbf{W}^i_{\text{base}})$
    end for
    Update the global prototype of each client with Eq. 3.11.
    Aggregate the client weights with averaging
    Send global prototypes to clients
    Send aggregated head and base models to clients
end for

LocalUpdate$(i, \mathbf{P}_i, \mathbf{W}^i_{\text{head}}, \mathbf{W}^i_{\text{base}})$:

for each local epoch **do**
    for batch $(\mathbf{x}, y) \in D_i$ **do**
        Freeze base, $\mathbf{W}^i_{\text{base}}$
        Update head, $\mathbf{W}^i_{\text{head}}$, with the loss in Eq. 3.2
    end for
end for
for each local epoch **do**
    for batch $(\mathbf{x}, y) \in D_i$ **do**
        Freeze head, $\mathbf{W}^i_{\text{head}}$
        Update base, $\mathbf{W}^i_{\text{base}}$, with the loss in Eq. 3.3
    end for
end for
Compute local prototype for each class as $\mathbf{p}^c_i = \frac{1}{N^c_i} \sum_{\mathbf{x}_j \in D^c_i} \mathbf{f}_i(\mathbf{x}_j)$
return $\mathbf{P}_i, \mathbf{W}^i_{\text{head}}, \mathbf{W}^i_{\text{base}}$

Figure 3.2. FedASA algorithm.

4. EXPERIMENTS

4.1. Dataset Distribution

To accurately simulate real-world scenarios in federated learning, where data distribution among clients is inherently non-IID (non-Independent and Identically Distributed), we employed the Dirichlet distribution to partition the datasets among multiple clients. This method is instrumental in reflecting the heterogeneity of client data, which poses significant challenges in deploying effective federated learning systems. A smaller value of the concentration parameter α, set to 0.1 for our experiments, generates substantial disparities in data distribution among the clients, enhancing the dataset's heterogeneity.

4.2. Datasets Used

Federated learning (FL) is a flexible and problem-independent methodology for decentralized machine learning, wherein data is retained across several devices or clients, therefore preserving privacy and minimizing dependency on centralized data storage. Recent years have seen substantial research on FL, particularly concentrating on its applications in supervised learning contexts. This is especially apparent in areas like image classification, where labeled datasets and deep learning models are frequently utilized.

Image classification is a widely utilized benchmark for assessing federated learning techniques, owing to its organized structure and the presence of recognized datasets like as CIFAR-10, CIFAR-100, FashionMNIST, and SVHN. These datasets include a variety of problems, such as class imbalance, heterogeneous data distributions (non-IID situations), and differing levels of data complexity, rendering them appropriate for evaluating the resilience and efficacy of FL methods. Utilizing these datasets, researchers can replicate realistic federated systems in which each client possesses access

solely to a portion of the data, mirroring real-world situations like mobile devices or edge computing. Consistent with previous research in this field, this study assesses the suggested FL methodology across multiple picture categorization tasks. This selection enables us to evaluate our approach against established baselines while investigating its performance over various data distributions, client engagement levels, and the extent of non-IID attributes.

Our focus on picture classification seeks to illustrate the versatility and adaptability of our method across various datasets and workloads, offering insights into its generalizability and scalability across decentralized machine learning environments.

(i) CIFAR-10 and CIFAR-100: These contain 60,000 32x32 color images across 10 and 100 classes, respectively. They provide a balanced multi-class classification problem, suitable for evaluating the effectiveness of federated learning algorithms across a diverse set of image recognition tasks.
(ii) SVHN (Street View House Numbers): Comprising digit images from Google Street View, this dataset allows us to test our models in a real-world numeric digit recognition scenario.
(iii) Fashion MNIST: With 70,000 images across 10 fashion product categories, Fashion MNIST offers a more challenging benchmark than the traditional MNIST digit dataset, making it ideal for more complex image recognition tests.

4.3. Client Configuration and Data Partitioning

We explored the impact of client participation on model training and performance through two specific case studies:

(i) Case 1: This case involved 20 clients with 100% participation in every training round, representing an ideal scenario with maximum client involvement.
(ii) Case 2: Comprising 100 clients with a participation rate of 30% per training round, this case reflects a more realistic scenario.

Data was distributed to clients via a non-IID methodology through the Dirichlet distribution, a commonly utilized technique in federated learning research. The Dirichlet distribution offers a systematic method for distributing data among clients, enabling researchers to model various levels of data heterogeneity by modifying the concentration parameter (α). This strategy is especially effective in generating realistic non-IID conditions, where data distributions among clients show significant diversity, replicating real-world conditions.

Following to this methodology ensures that our experimental configuration aligns to cutting-edge methods in federated learning, facilitating fair and important comparisons with current methodologies. This methodology has been extensively used in recent studies, including Hsu et al. [13] and Liang et al. [33], which emphasize its effectiveness in modeling heterogeneous data distributions for the assessment of federated learning algorithms. By implementing this standard technique, we establish a reliable and replicable framework for evaluating the success of our proposed strategy in varied and demanding federated situations.

4.4. Training Settings

We employ the SGD (Stochastic Gradient Descent) optimizer with a learning rate of 0.005 for all approaches. The value of the local training epochs is fixed to E = 1. The global communication rounds are set at a constant value of 200 for all datasets, as more communication rounds result in minimal or negligible gain in accuracy.

To assess the models' performance, we calculate the mean test accuracy among clients. In data with 20 clients, we used full join from clients, and in 100 client settings, we used a 30 percent join ratio from clients. A diverse set of libraries is used in the implementation of the project. PyTorch is mainly used for model training.

4.5. Compared Methods

Each baseline is trained and tested on the same data. This research builds on the implementation from *PFLlib: Personalized Federated Learning Algorithm Library* by Jianqing Zhang [34]. Each model are trained from scracth to be able to test them on same data sets with given conditions. We compare the following baselines:

- FedASA : Our method.
- FedProto: A prototype-based federated learning approach that aggregates prototypes for shared representation learning [27].
- FedBABU: A parameter decoupling method that decouples local updates into shared and personalized parameters [3].
- FedPAC: A personalized aggregation method that uses feature alignment and classifier collaboration [29].
- FedRep: Another parameter decoupling method that separates the global feature extractor and local classifiers [26].
- FedROD: Learns an additional local classifier based on the global feature extractor and uses model ensemble for prediction [28].

Table 4.1. Comparison of different methods on various datasets (values in percentages, with standard deviations in parentheses). All data sets have 0.1 α.

Method	Cifar100-100	Cifar100-20	Cifar10-100	Cifar10-20	FashionMNIST-20	FashionMNIST-100	SVHN-20
FedProto [27]	38.62 (9.6)	50.59 (5.0)	83.81 (10.7)	89.50 (14.3)	97.15 (9.7)	94.01 (4.61)	95.71 (10.3)
FedBABU [3]	45.09 (10.2)	53.81 (4.0)	86.98 (9.9)	89.73 (11.7)	97.11 (7.9)	95.07 (4.9)	96.16 (2.6)
FedPAC [29]	47.07 (9.2)	59.08 (3.6)	74.38 (11.5)	87.87 (23.8)	94.77 (15.9)	91.51 (20.6)	96.70 (2.3)
FedRep [26]	42.06 (9.8)	48.72 (4.2)	87.88 (9.7)	90.57 (10.6)	97.60 (6.0)	95.63 (4.2)	96.53 (2.3)
FedROD [28]	48.29 (9.1)	50.72 (4.0)	87.61 (9.6)	90.62 (11.2)	97.47 (5.7)	95.76 (4.1)	96.54 (2.3)
FedASA	48.27 (9.2)	57.38 (4.2)	**87.97 (10)**	**91.51 (8.7)**	**97.77 (5.7)**	95.73 (5.4)	**96.84 (2)**

This study compares the performance of different federated learning algorithms across several datasets, including CIFAR-100, CIFAR-10, FashionMNIST, and SVHN, to evaluate their effectiveness in solving issues like as data heterogeneity, personalization, and generalization.

Our suggested method, FedASA, shows improved adaptability and robustness across various data distributions, achieving results that are competitive with or superior to state-of-the-art methodologies as seen on Table 4.1. We reported overall accuracy across all clients at the end of the last communication round.

FedProto utilizes prototype-based federated learning to aggregate class-level prototypes from several clients. Although qualified with simpler datasets like FashionMNIST (0.9715 on FashionMNIST-20-01), it encounters difficulties with complex datasets such as CIFAR-100, achieving only 0.3862 on Cifar100-100-01. The fixed characteristics of prototype aggregation limit its adaptability in accepting diverse data. Conversely, FedASA adaptively customizes prototypes using a self-attention process, providing better results across all datasets.

FedBABU implements parameter decoupling to distinguish shared global parameters from client-specific modifications, resulting in consistent outcomes, including 0.4509 on Cifar100-100-01 and 0.9616 on SVHN-20-01. Nonetheless, it is less adaptable than FedASA, which utilizes self-attention for precise adjustments. For example, in Cifar10-20-01, FedASA outperforms FedBABU with a score of 0.9152 versus 0.8973.

FedPAC utilizes feature alignment and classifier collaboration, achieving a performance of 0.9477 on the FashionMNIST-20-01 dataset, however underperforming on the CIFAR-10 dataset with a score of 0.7438 on Cifar10-100-01. Conversely, FedASA reaches a significantly higher accuracy of 0.8814, demonstrating flexibility across many client contexts. FedRep emphasizes parameter decoupling by separating the global feature extractor from local classifiers, achieving impressive results on CIFAR-10, with scores of 0.8788 on Cifar10-100-01 and 0.9057 on Cifar10-20-01. Nonetheless, it shows poor results on difficult datasets such as CIFAR-100, achieving 0.4206 on CIFAR100-100-01 in contrast to FedASA's 0.4825. FedASA's dynamic prototype generation improves generalization and customization.

FedROD increases robustness through the integration of an additional local classifier and the application of model ensembles, reaching competitive outcomes of 0.4829 on Cifar100-100-01 and 0.9747 on FashionMNIST-20-01. Nonetheless, its ensemble methodology creates computational complexity, making it less scalable. FedASA achieves results that are either comparable or greater, specifically 0.9773 on FashionMNIST-20-01 and 0.9686 on SVHN-20-01, all while preserving a more straightforward and efficient design.

FedASA, our suggested methodology, appears by integrating federated learning with a self-attention mechanism to produce customized prototypes for each client. In Cifar100-100-01, achieved a score of 0.4825, outperforming FedProto, FedRep, and FedBABU. In Cifar10-100-01, reaches a score of 0.8814, exceeding all alternative approaches. In FashionMNIST-20-01, gets 0.9773, the highest score among all methodologies. In SVHN-20-01, reaches 0.9686, showing scalability and adaptability. FedASA enhances the balance between personalization and generalization by dynamically modifying client contributions through label-specific weights and self-attention, outperforming other methodologies. It offers a scalable and computationally efficient method that achieves state-of-the-art outcomes across datasets of differing complexity. These findings confirm FedASA as a strong and adaptable method for practical federated learning applications.

In addition to that perspective, number of parameters send at each method is also important metric to measure. In FedASA, We send model parameters, class prototypes and total number of samples of that client at each communication round. In FedRep and FedRod , only model parameters and client sample counts are sent to server. In FedBABU , model base and sample counts are sent to server. In FedProto, just prototypes are sent to server. Lastly, in FedPAC, they send model weights , sample counts of each user, prototypes and also 2 additional client's data distribution parameters to the server. Most of the cases the requirement for bandwidth are not significantly differ.

4.6. Ablation Study

To evaluate the performance and adaptability of the proposed federated learning framework, a series of experiments were conducted under different configurations and strategies. The focus was on understanding the impact of data heterogeneity, client-specific optimizations, and prototype-based methods on the overall system performance.

4.6.1. Prototype Distance for Head Aggregation FedDA

The first experiment explored the use of prototype distances to identify the closest clients in terms of their data distributions. The goal was to compute a weighted sum of the client-specific classifier heads based on these prototype distances and use the aggregated head to update the model on each client. The method involved:

- Prototype Distance Calculation: Measuring the similarity between client prototypes for each class.
- Head Aggregation: Computing a weighted average of the client's heads using the prototype similarity scores. These aggregated heads were then distributed back to the clients.

While the idea aimed to personalize the global model by leveraging the similarity of client data distributions, the approach did not yield satisfactory results. The aggregation mechanism introduced noise into the optimization process, leading to performance degradation rather than improvement. Consequently, this approach was not pursued further.

4.6.2. Self Attention Based Trials FEDASA

Those experiments are set of different approaches of mainly same idea. As we discussed earlier we proposed a novel approach to generate tailored global prototypes

for each client. But we tried some other implementations of it. First trial is generating one weight for each client then using that we calculated each client's prototype. In second approach, we changed the weights from one for each client to one for each client - label , so that every client can join other client's prototype by differing weights for each label. After that we also tried to change model parameters of self attention mechanism, by changing layer sizes for key and values using additional layers. Finally, we also tried to train attention mechanism in two settings. One we trained self attention from scratch at each server epoch (between communication rounds) and secondly we tried to train the self attention mechanism over all epochs. Below shows the results of those trials;

Table 4.2. Comparison of different methods on various datasets (values in percentages). All data sets have 0.1 α.

Method	Cifar100-100	Cifar100-20	Cifar10-100	Cifar10-20	FashionMNIST-20	FashionMNIST-100	SVHN-20
Fed_ASA_WA	47.28	57.00	88.13	91.62	97.81	95.67	96.91
Fed_ASA_LW	48.33	57.70	87.91	91.97	97.74	95.71	96.96
Fed_ASA_LD	46.24	57.56	88.21	91.72	97.73	95.63	96.87
FedASA	48.27	57.38	87.97	91.51	97.77	95.73	96.84

FedASAWA is the weight aggregation method, that We tried first, sum all the weights and then normalize all other clients' weights for given client in order to use weighted sum of prototypes. FedASALW is the label weight method, in this method we try to find weight for each label for all clients so that each client can contribute other clients prototypes by variying weights for different labels. FedASALD method is the larger layer method which try to change layer sizes and utilize a deeper model. FedASA method try to train self attention during all training procedure instead of training it from scracth at each server round. The training performance shown on the Table 4.2 based on average client accuracy on test set for each data.

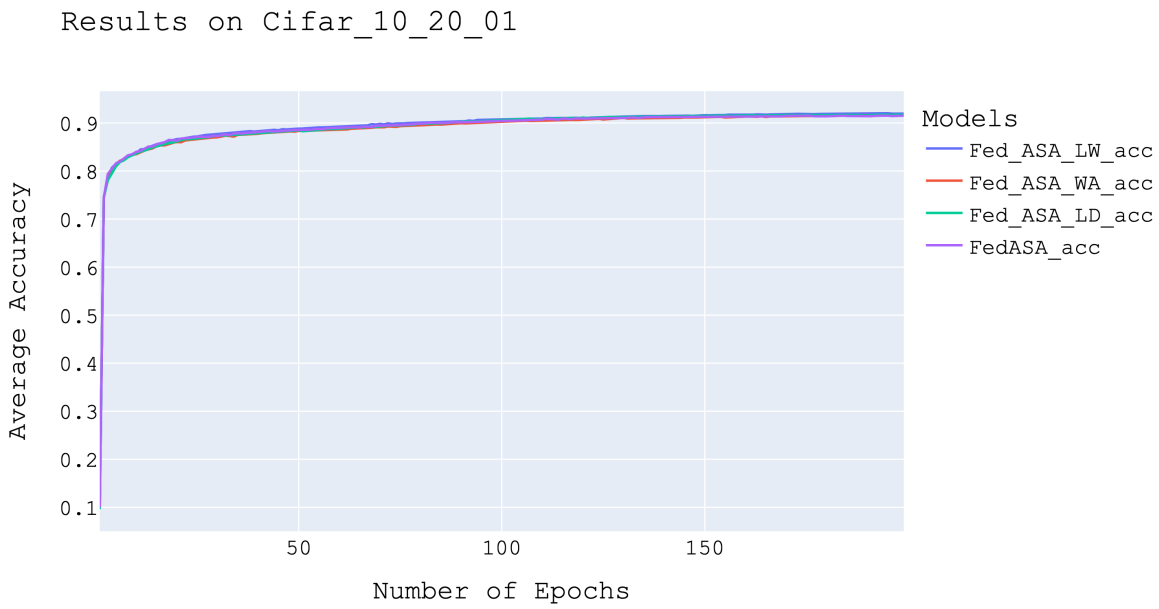

Figure 4.1. Performance comparison of FedASA variants on CIFAR-10 with 20 classes and $\alpha = 0.1$. All methods converge to similar accuracies, showcasing minimal variation for simpler datasets.

This graph shows the performance of FedASA variants on the Cifar10-20_01 dataset. All approaches demonstrate rapid convergence, with accuracy stabilizing around 90% after approximately 50 epochs. Among the four variants:

- Fed_ASA_LW (label weights) and FedASA (continuous self-attention training) maintain slightly higher average accuracy compared to the others.
- Fed_ASA_WA (weight aggregation) and Fed_ASA_LD (larger layers) are very close in performance, with minimal divergence.

The convergence is smooth and consistent across all of the approaches. Fine-tuning the attention mechanism through label-specific or continuous training improves accuracy slightly but does not produce drastic differences for this dataset.

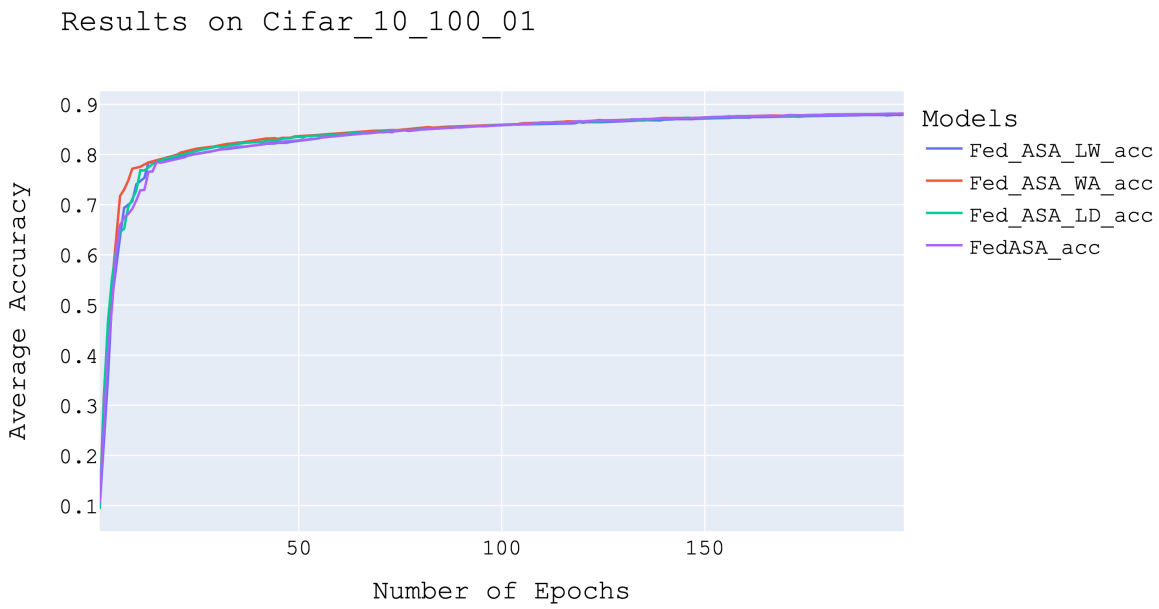

Figure 4.2. Performance comparison of FedASA variants on CIFAR-10 with 100 classes and $\alpha = 0.1$. The results show that Fed_ASA_LW and FedASA exhibit a slight performance advantage.

This graph demonstrates the results on Cifar10-100_01, where the task is more challenging due to the higher class complexity. Key observations include:

- Models experience slower convergence initially but stabilize around 85-87% accuracy after approximately 100 epochs.
- Fed_ASA_LW and FedASA achieve marginally better performance than Fed_ASA_WA and Fed_ASA_LD.
- The variance among models in earlier epochs is more noticeable here, highlighting the challenge of achieving optimal attention weight aggregation.

The results reflect the effectiveness of label-specific weights and continuous self-attention training under higher-class diversity scenarios.

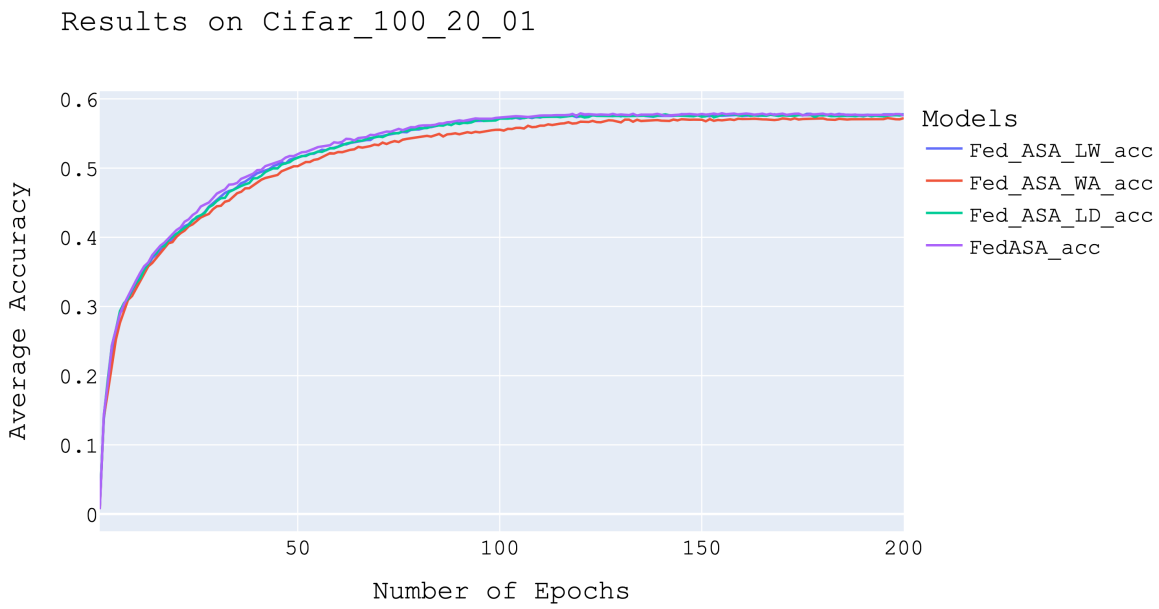

Figure 4.3. Performance comparison of FedASA variants on CIFAR-100 with 20 classes and $\alpha = 0.1$. Fed_ASA_LW and FedASA outperform other methods, highlighting the benefits of label-specific attention weights.

This graph illustrates performance on Cifar100-20_01, a more complex dataset with 100 classes grouped into 20 labels. Observations include:

- Accuracy begins around 30% and steadily increases, stabilizing at approximately 58% by 150 epochs.
- Fed_ASA_LW and FedASA consistently outperform other approaches, demonstrating that a tailored label-wise or continuous training approach is beneficial.
- Fed_ASA_WA slightly lags, indicating that a single weight per client is insufficient for handling class-specific nuances.

The results emphasize the importance of label-specific attention mechanisms when managing datasets with intricate hierarchical relationships.

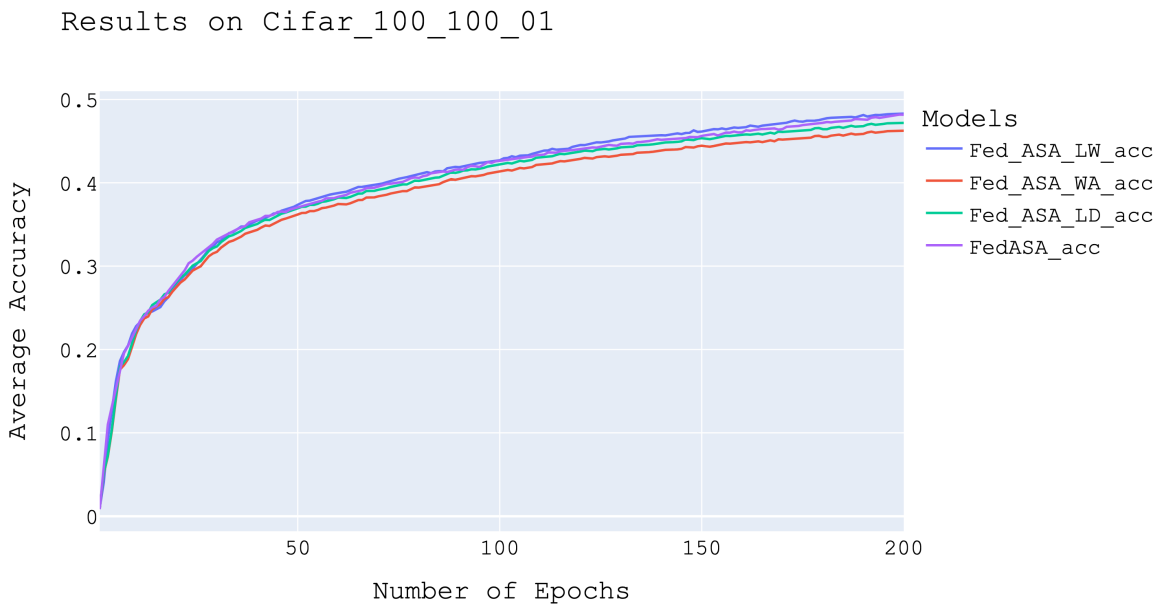

Figure 4.4. Performance comparison of FedASA variants on CIFAR-100 with 100 classes and $\alpha = 0.1$. Fed_ASA_LW achieves the highest accuracy, demonstrating its ability to handle complex, non-IID data.

For Cifar100-100_01, where all 100 classes are treated independently, the task is the most challenging. Key observations:

- Models start at a lower baseline (20% accuracy) and gradually rise to approximately 45% after 200 epochs.
- Fed_ASA_LW and FedASA achieve slightly better results in later epochs.
- Differences between the methods are minimal, indicating that dataset complexity saturates performance improvements.

This dataset highlights the need for more advanced strategies, as current methods converge to similar performance levels despite their sophistication.

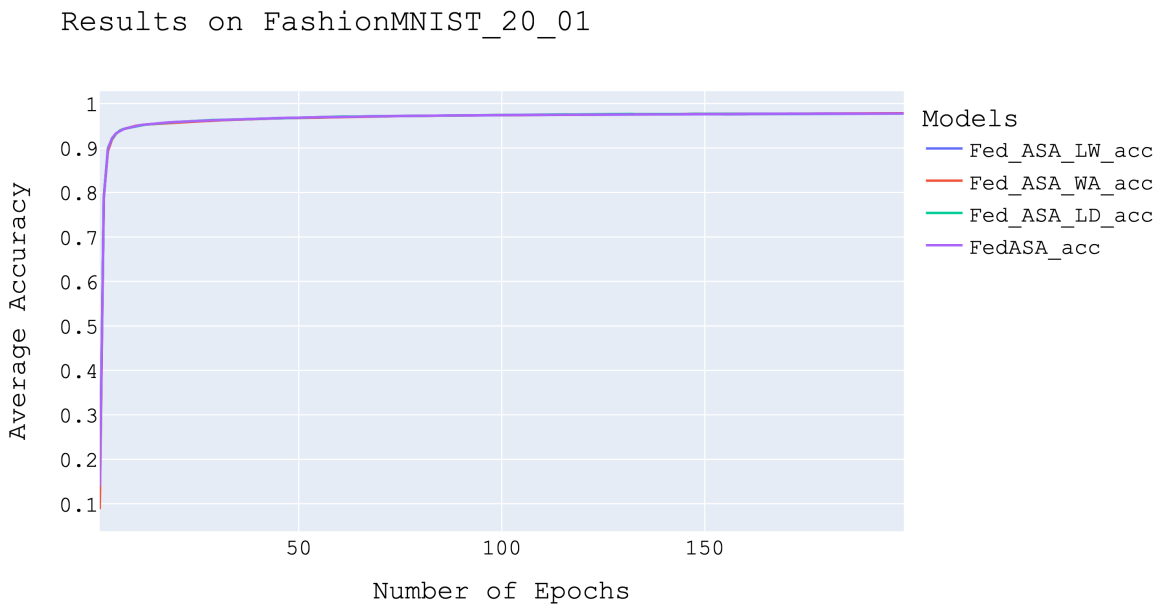

Figure 4.5. Performance comparison of FedASA variants on FashionMNIST with 20 classes and $\alpha = 0.1$. All methods converge rapidly, achieving near-optimal accuracy with minimal differences.

This graph presents results on the FashionMNIST-20_01 dataset, where all methods demonstrate nearly identical performance:

- Accuracy quickly reaches 95-97% within 50 epochs, stabilizing with minimal variation.
- FedASA achieves the highest accuracy, but the difference is negligible.
- Most of the models shows similar performances on FashionMNIST-20_01 dataset.

The simplicity of this dataset reduces the need for sophisticated self-attention mechanisms, as all methods converge to similar results. Additionally, convergence epochs are also nearly same for all methods.

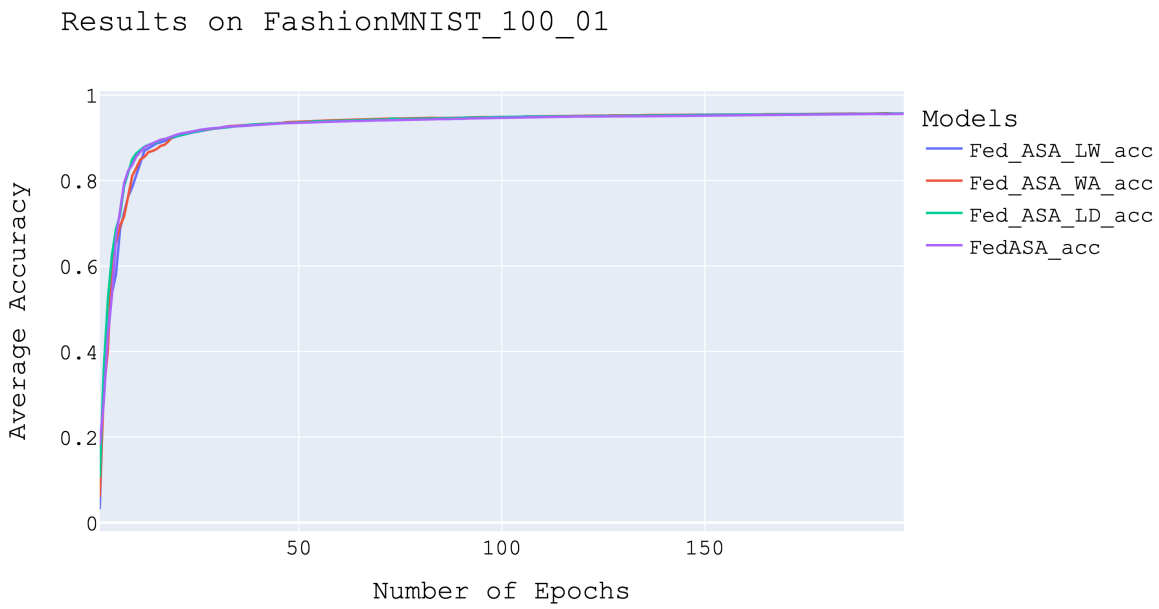

Figure 4.6. Performance comparison of FedASA variants on FashionMNIST with 100 classes and $\alpha = 0.1$. Variations among methods are negligible, as the dataset remains less complex.

In this experiment with FashionMNIST-100_01, the dataset complexity increases slightly:

- Accuracy stabilizes at 97-98% after 50 epochs, mirroring the pattern observed in the 20_01 setting.
- Fed_ASA_LW and FedASA perform marginally better, but differences remain minimal.

The dataset's structure does not significantly benefit from advanced methods, as all approaches achieve near-optimal accuracy. In earlier stages of epochs there is some difference between models but in the end of all of epochs most of the models converges to the same accuracy levels.

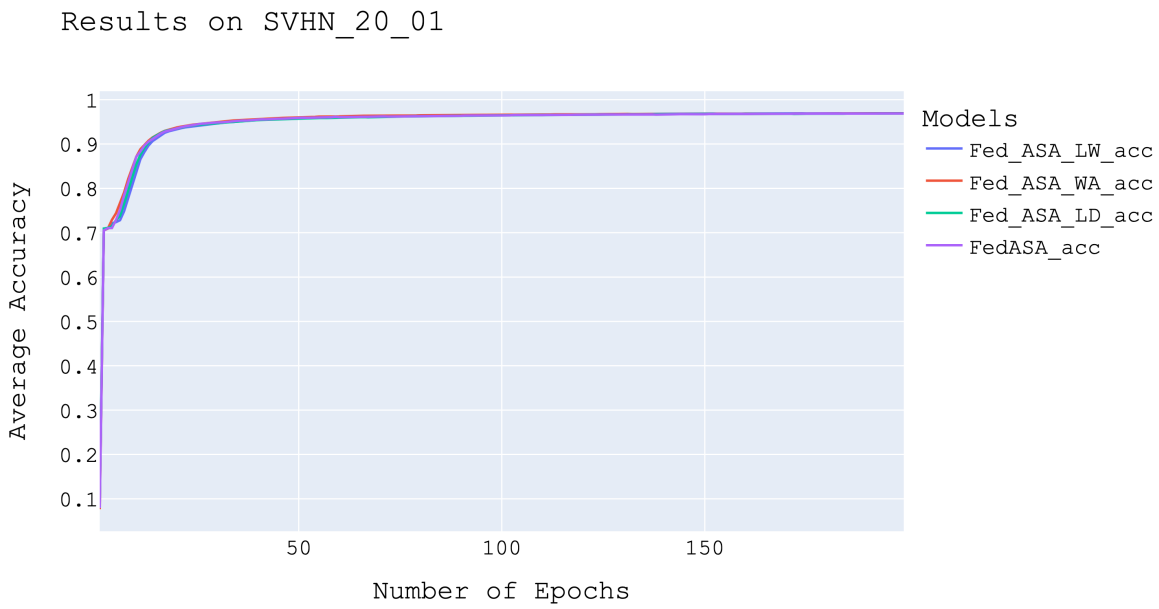

Figure 4.7. Performance comparison of FedASA variants on SVHN with 20 classes and $\alpha = 0.1$. The results show rapid convergence for all methods, with Fed_ASA_LW and FedASA maintaining slight performance advantages.

This graph captures performance on SVHN-20_01, a dataset with significant variability in image data:

- Models converge rapidly to 95-97% accuracy within 50 epochs.
- FedASA and Fed_ASA_LW demonstrate slight improvements, indicating the benefits of label-specific attention and continuous training.
- Fed_ASA_WA and Fed_ASA_LD perform closely but trail marginally behind.

Attention-based methods prove effective in handling datasets with natural variability, such as SVHN, though the differences remain subtle.

The experimental results across multiple datasets, including CIFAR-10, CIFAR-100, FashionMNIST, and SVHN, provide valuable insights to the performance of the proposed FedASA variants. The following key observations can be drawn:

(i) Label-Specific Attention Weights (Fed_ASA_LW):

- This approach consistently gives higher accuracy across most of the datasets, especially on CIFAR-100 and complex settings with high class diversity.
- Assigning label-specific weights allows clients to contribute to other clients' prototypes in a more detailed way, improving the alignment between local and global prototypes.

(ii) Continuous Self-Attention Training (FedASA):

- Training the self-attention mechanism throughout the entire training procedure (instead of restarting at each communication round) results in improved accuracy, such as (CIFAR-100) which is more complex data. However, for simpler datasets like FashionMNIST, the performance gain is negligible, indicating diminishing returns for over-parameterized mechanisms.

(iii) Weight Aggregation (Fed_ASA_WA):

- While the weight aggregation approach is computationally efficient, its performance is limited in datasets with non-IID data or high class complexity, such as CIFAR-100.
- This method works well for simpler tasks, but it does not effectively capture client-specific details.

(iv) Larger Layer Design (Fed_ASA_LD):

- Increasing the self-attention layer size improves performance in complex scenarios; however, the gains are less significant compared to Fed_ASA_LW and FedASA. The additional computational overhead might not always justify the small improvement in accuracy.

(v) Dataset Complexity and Convergence Patterns:

- For simpler datasets such as FashionMNIST and SVHN, all methods converge rapidly to near-optimal accuracy, demonstrating minimal variation between approaches. In contrast, for more complex datasets such as CIFAR-100, label-specific attention and continuous self-attention training provide important improvements, as these methods better address data heterogeneity.

The experiments highlight that methods using label-specific attention weights (Fed_ASA_LW) and continuous training of the self-attention mechanism (FedASA) are more effective in addressing data heterogeneity and improving performance. Simpler methods like weight aggregation (Fed_ASA_WA) and larger layer design (Fed_ASA_LD) are beneficial in specific experiments but fail to generalize as effectively across all datasets.

5. CONCLUSION

This thesis conducted a comprehensive investigation of federated learning techniques, focusing on the difficulties of data heterogeneity, personalization, and scalability in decentralized settings. We introduced a novel method, FedASA, which utilizes a self-attention mechanism to dynamically create customized global prototypes for each client. FedASA addresses the difference between global generalization and local personalization through the implementation of label-specific weighting and continuous learning techniques, offering a strong solution for practical federated learning applications.

Our experimental findings indicate that FedASA either outperforms or equals the performance of leading approaches, including FedProto, FedPAC FedBABU, and FedRep, across a variety of datasets, such as CIFAR-100, CIFAR-10, FashionMNIST, and SVHN. The dynamic prototype generation and self-attention mechanism allowed FedASA to effectively adapt to non-IID data distributions, resulting in enhanced accuracy and consistency across both complex and simple datasets. The method's capacity to scale effectively while ensuring computational feasibility underlines its practical significance.

This study assessed various modifications and enhancements to the suggested system, encompassing label-specific weighting, architectural modifications, and diverse training strategies for the self-attention mechanism. These investigations provided significant insights into the trade-offs among computing complexity, flexibility, and performance, defining a framework for future improvements in federated learning.

This thesis presents innovative methodologies for personalized federated learning (FL) that tackle issues caused by data heterogeneity and client-specific requirements. Although our methodologies have shown considerable enhancements in accuracy and adaptation on benchmark datasets, many possible options for further research exist.

A primary focus is the implementation of these methods on larger and more heterogeneous datasets. Evaluating the proposed methodologies in practical applications, such as healthcare or financial systems, will help in verifying their robustness and generalization.

Another significant domain for future investigation is how to adapt of these methodologies to dynamic situations. In practical federated learning environments, clients frequently enter or exit the network and encounter variations in data distributions across time. Incorporating dynamic involvement and data drift through online learning or meta-learning techniques could improve the practical usability of federated learning. Moreover, although the proposed approaches enhance performance, they incur some computational and communication overheads. The next steps could focus on enhancing the efficiency of these systems by the application of techniques such as model compression, less communication, or light weight architectures for self attention training.

REFERENCES

1. Nguyen, D. C., M. Ding, P. N. Pathirana, A. Seneviratne, J. Li and H. V. Poor, "FederatedLearning for Internet of Things: A Comprehensive Survey", *IEEE Communications Surveys & Tutorials*, Vol. 23, No. 3, pp. 1622–1658, 2021.

2. Zhang, C., Y. Xie, H. Bai, B. Yu, W. Li and Y. Gao, "A Survey on Federated Learning", *Knowledge-Based Systems*, Vol. 216, p. 106775, 2021.

3. Oh, J., S. Kim and S.-Y. Yun, "FedBABU: Toward Enhanced Representation for Federated Image Classification", *International Conference on Learning Representations*, Online, 2022.

4. McMahan, B., E. Moore, D. Ramage, S. Hampson and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data", *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, Vol. 54 of *Proceedings of Machine Learning Research*, pp. 1273–1282, Ft. Lauderdale, 2017.

5. Briggs, C., Z. Fan and P. Andras, "Federated Learning with Hierarchical Clustering of Local Updates to Improve Training on Non-IID Data", *International Joint Conference on Neural Networks*, pp. 1–9, Glasgow, 2020.

6. Mansour, Y., M. Mohri, J. Ro and A. T. Suresh, "Three Approaches for Personalization with Applications to Federated Learning", ArXiv preprint arXiv:2002.10619, 2020.

7. Chen, Y., X. Qin, J. Wang, C. Yu and W. Gao, "Fedhealth: A Federated Transfer Learning Framework for Wearable Healthcare", *IEEE Intelligent Systems*, Vol. 35, No. 4, pp. 83–93, 2020.

8. Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u.

Kaiser and I. Polosukhin, "Attention is All You Need", *Advances in Neural Information Processing Systems*, Vol. 30, pp. 6000–6010, Long Beach, 2017.

9. Zhao, Y., M. Li, L. Lai, N. Suda, D. Civin and V. Chandra, "Federated Learning with Non-IID Data", ArXiv preprint arXiv:1806.00582, 2018.

10. Li, T., A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar and V. Smith, "Federated Optimization in Heterogeneous Networks", *Proceedings of Machine Learning and Systems*, Vol. 2, pp. 429–450, Austin, 2020.

11. Acar, D. A. E., Y. Zhao, R. M. Navarro, M. Mattina, P. N. Whatmough and V. Saligrama, "Federated Learning Based on Dynamic Regularization", *International Conference on Learning Representations*, Vienna, 2021.

12. Karimireddy, S. P., S. Kale, M. Mohri, S. Reddi, S. Stich and A. T. Suresh, "Scaffold: Stochastic Controlled Averaging for Federated Learning", *International Conference on Machine Learning*, Vol. 119, pp. 5132–5143, Vienna, 2020.

13. Hsu, T.-M. H., H. Qi and M. Brown, "Measuring The Effects of Non-Identical Data Distribution for Federated Visual Classification", ArXiv preprint arXiv:1909.06335, 2019.

14. Yu, H., N. Zhang, S. Deng, Z. Yuan, Y. Jia and H. Chen, "The Devil is The Classifier: Investigating Long Tail Relation Classification with Decoupling Analysis", ArXiv preprint arXiv:2009.07022, 2020.

15. Wang, J., V. Tantia, N. Ballas and M. Rabbat, "SlowMo: Improving Communication-Efficient Distributed SGD with Slow Momentum", *International Conference on Learning Representations*, Addis Ababa, 2020.

16. Lin, T., L. Kong, S. U. Stich and M. Jaggi, "Ensemble Distillation for Robust Model Fusion in Federated Learning", Vol. 33, pp. 2351–2363, Online, 2020.

17. Li, D. and J. Wang, "Fedmd: Heterogenous Federated Learning via Model Distillation", ArXiv preprint arXiv:1910.03581, 2019.

18. Zhu, Z., J. Hong and J. Zhou, "Data-Free Knowledge Distillation for Heterogeneous Federated Learning", *International Conference on Machine Learning*, Vol. 139, pp. 12878–12889, Online, 2021.

19. Smith, V., C.-K. Chiang, M. Sanjabi and A. Talwalkar, "Federated Multi-Task Learning", ArXiv preprint arXiv:1705.10467, 2017.

20. Dinh, C. T., T. Tran and T. Nguyen, "Personalized Federated Learning with Moreau Envelopes", *Advances in Neural Information Processing Systems*, Vol. 34, pp. 21394–21407, Online, 2021.

21. Yang, H., H. He, W. Zhang and X. Cao, "Fedsteg: A Federated Transfer Learning Framework for Secure Image Steganalysis", *IEEE Transactions on Network Science and Engineering*, Vol. 8, No. 2, pp. 1084–1094, 2020.

22. T.Dinh, C., N. Tran, T. D. Nguyen, W. Bao, A. Zomaya and B. Zhou, "Federated Learning with Proximal Stochastic Variance Reduced Gradient Algorithms", *International Conference on Parallel Processing*, pp. 1–11, Edmonton, 2020.

23. Li, T., S. Hu, A. Beirami and V. Smith, "Ditto: Fair and Robust Federated Learning Through Personalization", *International Conference on Machine Learning*, pp. 6357–6368, Online, 2021.

24. Arivazhagan, M. G., V. Aggarwal, A. K. Singh and S. Choudhary, "Federated Learning With Personalization Layers", ArXiv preprint arXiv:1912.00818, 2019.

25. Zhuang, W., Y. Wen, X. Zhang, X. Gan, D. Yin, D. Zhou, S. Zhang and S. Yi, "Performance Optimization of Federated Person Re-Identification via Benchmark Analysis", *Proceedings of the 28th ACM International Conference on Multimedia*, pp. 955–963, 2020.

26. Collins, L., H. Hassani, A. Mokhtari and S. Shakkottai, "Exploiting Shared Representations for Personalized Federated Learning", *International Conference on Machine Learning*, pp. 2089–2099, Online, 2021.

27. Tan, Y., G. Long, L. Liu, T. Zhou, Q. Lu, J. Jiang and C. Zhang, "Fedproto: Federated Prototype Learning Across Heterogeneous Clients", *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36, pp. 8432–8440, Online, 2022.

28. Chen, H.-Y. and W.-L. Chao, "On Bridging Generic and Personalized Federated Learning for Image Classification", *International Conference on Learning Representations*, Online, 2022.

29. Jian Xu, S.-L. H., Xinyi Tong, "Personalized Federated Learning with Feature Alignment and Classifier Collaboration", *International Conference on Learning Representations*, Kigali, 2023.

30. Jianqing Zhang, Y. H. J. C., Yang Liu, "FedTGP: Trainable Global Prototypes with Adaptive-Margin-Enhanced Contrastive Learning for Data and Model Heterogeneity in Federated Learning", *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38, No. 15, pp. 16768–16776, 2024.

31. Zhang, J., Y. Liu, Y. Hua and J. Cao, "An Upload-Efficient Scheme for Transferring Knowledge From a Server-Side Pre-trained Generator to Clients in Heterogeneous Federated Learning", *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12109–12119, Seattle, 2024.

32. Li, H., Z. Cai, J. Wang, J. Tang, W. Ding, C.-T. Lin and Y. Shi, "Fedtp: Federated Learning by Transformer Personalization", *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 35, No. 10, pp. 13426 – 13440, 2023.

33. Liang, P. P., T. Liu, L. Ziyin, N. B. Allen, R. P. Auerbach, D. Brent, R. Salakhutdinov and L.-P. Morency, "Think Locally, Act Globally: Federated Learning with Local and Global Representations", ArXiv preprint arXiv:2001.01523, 2020.

34. Zhang, J., “PFLlib: Personalized Federated Learning Algorithm Library”, 2024, https://github.com/TsingZ0/PFLlib, accessed on May 11,2024.