Crossing Domain Borders with Federated Few-Shot Adaptation

Manuel Röder[1][a], Maximilian Münch[2][b], Christoph Raab[3][c] and Frank-Michael Schleif[1][d]
[1]*Faculty of Computer Science and Business Information Systems, Technical University of Applied Sciences Würzburg-Schweinfurt, Würzburg, Germany*
[2]*Department of Computer Science, University of Groningen, Groningen, Netherlands*
[3]*IAV GmbH, Berlin, Germany*

Keywords: Federated Learning, Domain Adaptation, Few-Shot Learning, Deep Transfer Learning, Resource Constraints, Sporadic Model Updates.

Abstract: Federated Learning has gained significant attention as a data protecting paradigm for decentralized, client-side learning in the era of interconnected, sensor-equipped edge devices. However, practical applications of Federated Learning face three major challenges: First, the expensive data labeling process required for target adaptation involves human participation. Second, the data collection process on client devices suffers from covariate shift due to environmental impact on attached sensors, leading to a discrepancy between source and target samples. Third, in resource-limited environments, both continuous or regular model updates are often infeasible due to limited data transmission capabilities or technical constraints on channel availability and energy efficiency. To address these challenges, we propose *FedAcross*, an efficient and scalable Federated Learning framework designed specifically for real-world client adaptation in industrial environments. It is based on a pre-trained source model that includes a deep backbone, an adaptation module, and a classifier running on a powerful server. By freezing the backbone and the classifier during client adaptation on resource-constrained devices, we enable the domain adaptive linear layer to solely handle target domain adaptation and minimize the overall computational overhead. Our extensive experimental results validate the effectiveness of *FedAcross* in achieving competitive adaptation on low-end client devices with limited target samples, effectively addressing the challenge of domain shift. Our framework effectively handles sporadic model updates within resource-limited environments, ensuring practical and seamless deployment.

1 INTRODUCTION

Traditional machine learning requires a centralized data center to store and aggregate collected training data as obtained from local devices, such as mobile phones, drones or thin clients. This approach has proven to be impractical for real-world application areas, as it requires considerable effort to collect and label data from different sources in compliance with data protection regulations.

In (McMahan et al., 2017) *Federated Learning* (FL) was introduced as a means of mitigating the security risks and costs associated with the implementation of traditional models. The proposed architecture enables multiple edge devices to jointly learn a global machine learning model under the administration of a central server, while local data stay with the client. Recent years have witnessed remarkable advancements in hardware and software technologies, with notable growth observed in the proliferation and interconnection of sensor-equipped edge devices (Siqueira and Davis, 2021), nowadays frequently employed in industrial production environments. This development, coupled with the increasing use of 5G-capable end devices, has significantly boosted the attractiveness of FL for practical industry applications and research purposes (Hard et al., 2018; Yang et al., 2018; Yang et al., 2019; Yang et al., 2020).

A popular real-world use case, where the typical FL approach is not directly applicable, is the classification of waste items (Laier and Laier, 2023; Bashkirova et al., 2023) by resource-constrained client devices equipped with visual sensors located at different waste sorting facilities (see conceptual design shown in Figure 1). In times of constantly

[a] https://orcid.org/0009-0003-4907-3999
[b] https://orcid.org/0000-0002-2238-7870
[c] https://orcid.org/0000-0002-6988-353X
[d] https://orcid.org/0000-0002-7539-1283

Röder, M., Münch, M., Raab, C. and Schleif, F.
Crossing Domain Borders with Federated Few-Shot Adaptation.
DOI: 10.5220/0012351900003654
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 13th International Conference on Pattern Recognition Applications and Methods (ICPRAM 2024)*, pages 511-521
ISBN: 978-989-758-684-2; ISSN: 2184-4313
Proceedings Copyright © 2024 by SCITEPRESS – Science and Technology Publications, Lda.

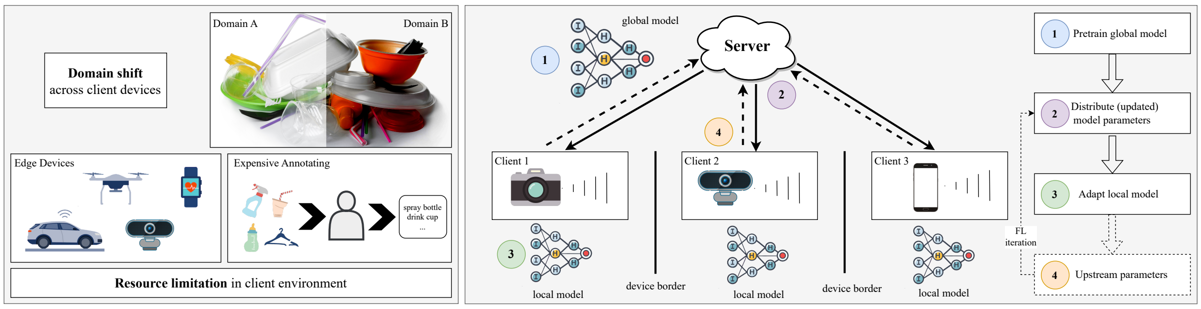

Figure 1: Challenges (left) and proposed approach (right) for federated client adaptation.

increasing amounts of garbage, waste sorting is a crucial challenge in many communities and intelligent sensor systems are a key element in the different strategies to address the problem (Lange, 2021). In this scenario, each client's local model is trained using isolated processing units, while data is generated locally and remains decentralized. *Cross-device federated learning* (Kairouz et al., 2021) tackles the aforementioned challenges of collaborative learning across multiple machines with limited data collection, expensive labeling and restricted sharing. Furthermore, there are several external drivers influencing the availability and quality of relevant sensor information. Considering e. g. visual sensors, domain shifts (see Figure 1, left) in captured images may occur due to variations in lightning or environmental conditions that affect brightness, contrast, color temperature, perspective, and noise (Koh et al., 2021). In general, the challenge of adjusting a system trained in the source domain to perform better in the target domain is referred to as *source-to-target adaptation* (Raab et al., 2022). Considering the limited hardware resources of clients, such as computing power, transmission capacity, and memory consumption, it is essential to design a training algorithm that minimizes client-side load while maintaining accurate model training. Consequently, this work focuses on integrating an established pre-train and fine-tune strategy into FL, aiming to transform domain-specific features into a task-invariant metric space to mitigate the effects of domain shift under resource constrains.

To address transmission costs and privacy restrictions (see Section 3.5 for details) in cross-device FL, we provide a practical solution not considered by now in FL across multiple domains based on *prototypes* (Snell et al., 2017), reducing data transfer overhead and minimizing computational costs for client device inference: Local prototypes are computed as class-wise averaged feature vectors for memory efficiency and client-side label prediction is carried out by comparing the distances between projected inputs and class prototypes for CPU usage optimization. As detailed below, this article presents a novel FL framework *FedAcross*, addressing real-world challenges of data sparsity on isolated clients as well as domain shift occurring across clients deployed in unrestrained industrial settings.

Main Contributions.

1. A computation-efficient FL approach is proposed to tackle target adaptation issues with limited labeled samples and distributional shifts across siloed devices in real-world applications. The outlined concept aims to improve per-device local models on downstream classification tasks while iteratively optimizing global model parameters to enhance bootstrapping of new FL clients.
2. We provide a ready-to-deploy, efficient and highly scalable, end-to-end FL solution based on Pytorch Lightning (Falcon and team, 2019) and Flower (Beutel et al., 2020), available on Github[1].
3. We thoroughly assess our method on a waste item classification scenario using domain adaptation benchmark data sets mirroring production conditions and observed competitive adaptation performance to state-of-the-art methods. This scenario is exceptionally challenging and covers a broad spectrum of real-life particularities in cross-device FL. Notably, our approach is not limited to waste item classification but can be effectively applied to various other use cases[2], highlighting its versatility and potential.

2 RELATED WORK

Previous studies have extensively examined the limitations of device-based FL, in which data remains isolated within individual entities or organizations. Researchers have conducted investigations to address the

[1]https://github.com/cairo-thws/FedAcross

[2]Supply chain optimization, smart grid energy management, epidemic and disease surveillance

challenges associated with this siloed approach, including data privacy concerns, communication overhead, and model performance degradation (Zhao et al., 2022). However, the existing approaches still have certain limitations in efficiently utilizing the information present in siloed data while maintaining privacy. Achieving efficient information transfer and compact encoding is crucial to overcome the barriers of crossing device borders with minimal transmission costs and achieving robust generalization capabilities (Zhao et al., 2022).This is getting even more challenging in realistic scenarios, where often only weakly labeled data are available.

Techniques addressing *centralized* scenarios have been proposed using *few-shot learning* (FSL) to effectively enable models to learn from limited labeled data (Wang et al., 2020; Song et al., 2023; Hu et al., 2022). These approaches usually rely on fine-tuning strategies to improve the model's ability to generalise and adapt to new tasks, even when only a few labeled training samples are available. Two main training paradigms have emerged: *learning by transfer* (Dhillon et al., 2020), where a deep neural network is trained on a source data set and subsequently fine-tuned on a downstream few-shot learning task, and *meta-learning* approaches see e.g. (Snell et al., 2017), where incremental parameter updates encode task-specific background information in the model optimization process.

Additionally, the siloed setting and common phenomena of error-prone measurement devices introduce various effects of cross-domain shifts. Cross-domain learning specifically focuses on transferring knowledge from a source domain to a target domain, even when the data's characteristics or distribution significantly differ. Extensive research in this area has explored *domain adaptation* (DA) techniques such as domain alignment, feature mapping and instance reweighting to improve model performance when applied to unseen target domains (Raab et al., 2022)

Moreover, *test-time adaptation methods* (Nado et al., 2021; Zhang et al., 2022) require only few labeled data points per class from a target domain to optimize domain-specific adaptation parameters, thereby providing an effective blueprint to compose our model architecture.

Cross-domain few-shot learning (CD-FSL) deals with a centralized combination of the aforementioned challenges, namely the effective and fast learning of relevant information with only a few samples while coping with the distributional shift between source and target data. Recent studies on CD-FSL (Guo et al., 2020) have shown that transfer learning approaches outperform state-of-the-art meta-learning methods on FSL benchmarks over multiple domains. Therefore, transfer learning represents a reasonable solution to not only avoid computationally intensive gradient update calculations for client models running on edge devices, but also to outsource the heavy lifting of training a source model on a large data set to a well-equipped server instance. With FedProto (Tan et al., 2022), which is a prototype-based aggregation method for heterogeneous clients and FPL (Huang et al., 2023), which constructs server-side cluster prototypes, *prototypes* have already been used in FL contexts; however, it does not address client-side resource limitations and requires many labeled data. Our approach combines ideas from different fields addressing the mentioned issues to provide a practical solution.

3 METHODOLOGY

In Section 3.2 we provide the main prerequisites, followed by an overview of the server model architecture. An end-to-end source model training procedure as well as some theoretical background is provided in Section 3.1. Section 3.3 examines the on-client model adaptation process under the constraints of minimal availability of labeled samples and domain shift. Subsequently we introduce a computation-efficient, prototype-based client inference pipeline in Section 3.4 followed by the pseudo-code for *FedAcross* and prototype upstreaming options in Section 3.5.

3.1 Server Model Training

The proposed server model consists of four main components as visualized in Figure 2. Throughout this chapter, we describe in detail each of these components and deliver a justification of the respective design choice in regards to the proclaimed challenges.

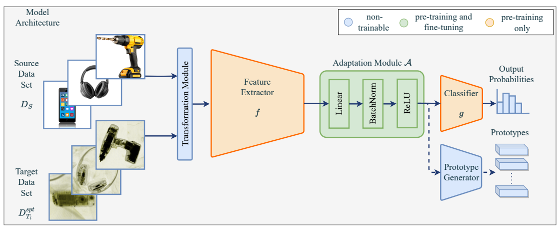

Figure 2: Model Architecture for server pre-training on source data set D_S and client fine-tuning on target data set $D_{T_i}^{spt}$. Module colors determine whether the parameters are frozen or trainable during respective training stages.

3.2 Prerequisites

We consider a publicly available, labeled source data set $D_S = \{X_S, Y_S\} = \{x_l, y_l\}_{l=1}^{N} \overset{i.i.d.}{\sim} p_S(X_S, Y_S)$ in the source domain S distributed by p_S. For each client i we have a target data set $D_{T_i} \overset{i.i.d.}{\sim} \{x_l, y_l\}_{l=1}^{k_i} \overset{i.i.d.}{\sim} p_{T_i}(X_{T_i}, Y_{T_i})$ in the target domain T_i. Without loss of generality, we assume the amount of labeled samples per class $k = k_i = \{0, 3, 5, 10\}$ for client i. Samples are given as $x_l \in \mathbb{R}^d$, d as number of features and $y_l \in Y$, Y as a discrete label space, common for source and target domain, $|Y| = L$. The distributions $p_S(X_S, Y_S) \neq p_{T_i}(X_{T_i}, Y_{T_i})$ are subject to domain shift. Similar as x_l, we introduce class-wise prototypes $\omega^{(n)}$ for the different source and target domains, where n is a class index in Y. A central server handles the bulk of model learning, while the training data is stored in separate silos at different clients with limited communication and processing power. The objective is to train a classifier model that can generalize to related target domains (see ”DomainA” and ”DomainB” in Figure 1 as an example for source and target domain differences). With $f(\phi)$ being a feature extractor parameterized by ϕ and $\mathcal{A}(\psi)$ an adaptation module parameterized by ψ. Eventually a classifier $g(\nu)$ is generated, parameterized by ν. The approach is applicable to various tasks, although we assume the feature space is related to image processing. Given a FL setup as seen in Figure 1, a computationally powerful server instance can fully access the source domain data set D_S. Additionally, each client i participating in the distributed learning process has exclusive access to its target domain data set D_{T_i}. To reflect real-world conditions in the modeling process, the following assumptions are also made:

- The number of annotated samples per class in the target domain data set D_{T_i} is fixed by k. The k-shot support set of client i is then denoted with $D_{T_i}^{spt} \subset D_{T_i}$, resulting in input-output pairs within each observed data set being equally distributed. In production, operators collect and annotate only k samples for local model fine-tuning.
- The number of classes used for local fine-tuning can be limited to n for each client separately, giving the client operator the option to individually select a subset of available classes to meet their needs.

In the following paragraphs we propose a model architecture that adheres to the above constraints and addresses the challenges presented from beginning to end. A summary including the most important notation used in this work can be found in Table 1.

The *transformation module* $\mathcal{T}$ is the first component of the server model. It maps the input x_S taken from the source domain data set D_S onto the output $\tilde{x}_S$ by applying a single augmentation chain that deploys concatenated affine transformations to alter model input data samples using $\tilde{x}_S \leftarrow \mathcal{T}(x_S)$. Affine data augmentation is a widely used and valid tool to avoid overfitting in the context of image classification using deep learning models (Perez and Wang, 2017). Further studies (Kim et al., 2022) revealed these data augmentation techniques are also beneficial for CD-FSL to increase the data set size as well as to improve the training procedure on the transfer learning downstream task on the target data set. Since the server model is trained on the entire source data set D_S, we follow their base augmentation method for *full fine-tuning* scenarios, where the entire network parameters are refreshed, and deploy horizontal flipping, random resized cropping and color jittering into the pipeline of $\mathcal{T}$.

Table 1: Notation Summary.

Notation	Description
D_S	source domain data set on server S
D_{T_i}	target domain data set on client i
$D_{T_i}^{spt}$	k-shot support set on client i, $D_{T_i}^{spt} \subset D_{T_i}$
N	nr. of observed classes, $n = 1, ..., N$
K	nr. of labeled samples per class, $k = 0, ..., K$
$f(\phi)$	feature extractor f parameterized by ϕ
$\mathcal{A}(\psi)$	adaptation module $\mathcal{A}$ parameterized by ψ
$g(\nu)$	classifier g parameterized by ν
$\omega_i^{(n)}$	prototype of class n on client i

The *feature extractor* f, parameterized by ϕ_S, is the second model component that retrieves relevant features from the transformed input data $\tilde{x}_S$ and produces output data $\mathring{x}_S$, with $f(\tilde{x}_S) \mapsto \mathring{x}_S$, $\mathring{x}_S \in \mathbb{R}^m, m \ll d$. As a compromise between the network depth required for the image classification problem and the constraint of keeping the run-time resource usage as low as possible for client endpoints in the fine-tuning stage later on, a pre-trained ResNet-34 (He et al., 2016) backbone with about 22 million parameters is employed on server side.

Following the feature extractor f, the next component of the source model is the *adaptation module* $\mathcal{A}$ parameterized by ψ_S. This is the core of our contribution to this work being built upon the concept of *task-specific adapters* (Li et al., 2022) and *universal templates for few-shot learning* (Triantafillou et al., 2021): a domain-adaptive linear layer allocates a dedicated set of conditional batch normalization parameters and linear layer weights for the source domain pre-training and each downstream target fine-tuning task, rendering the adaptation module $\mathcal{A}$ being fully responsible for DA (Chang et al., 2019). Formally, the server-side adaptation module is defined as

$$\begin{aligned}\mathcal{A}(\psi) &= \mathcal{A}(\mu,\sigma,\gamma,\beta,W,b;\overset{\circ}{x}_S)\\ &= \frac{(W\overset{\circ}{x}_S + b) - \mu}{\sigma}\gamma + \beta\end{aligned} \tag{1}$$

with $\overset{\circ}{x}_S$ denoting the output of the feature extractor, $\{\mu,\sigma\}$ denoting the batch norm statistics, $\{\gamma,\beta\}$ are the batch norm parameters and $\{W,b\}$, $W \in \mathbb{R}^{m\times m}$, $b \in \mathbb{R}$ are the weights and bias parameters of the linear layer, respectively. Parameter simplifications are possible by taking dependencies into account. In the server model, the *classification head* g is realized by a fully connected layer that receives the output of the adaptation module and projects that output to a specific set of labels. Following the definition of all essential components of the source model, the overall decision function is expressed as $\mathcal{F}(\phi_S,\psi_S,\nu_S) = g(\nu_S)\circ\mathcal{A}(\psi_S)\circ f(\phi_S)$. The training objective for our server-side classification task is defined as

$$\underset{\phi_S,\psi_S,\nu_S}{\arg\min}\, \mathcal{L}_{CE}(\mathcal{F}(\phi_S,\psi_S,\nu_S;\tilde{x}_S),y_S) \tag{2}$$

where $\mathcal{L}_{CE}$ is the cross entropy loss regularized by label smoothing (Müller et al., 2019) to further encourage robust output features and y_S the ground-truth label associated with $\tilde{x}_S$. All optimizations are done with stochastic gradient descent + momentum until convergence.

Overall, server pre-training on the source domain data set D_S is intended to learn and refine a set of features that are both discriminatory and transferable, thereby mitigating the difference between source and target domains. This results in the adapted parameters of the server model being used as initial parameters for client devices joining the FL process. Technically, the parameter transmission can be performed in two ways:

- **On Demand.** Upon joining the FL process for the first time, the central server transmits model parameters to the client. The disadvantage of this method is the high communication costs associated with the initial transfer of all parameters, despite it being a flexible solution.
- **Pre-Configured.** Upon installation, the client includes a pre-trained model parameter configuration, thereby eliminating the need to update the weights at the start and reducing the amount of communication involved.

Further, low-end client devices join the FL process of the central server step by step and follow the adaptation as described in Section 3.3.

3.3 Client Adaptation

For each client the respective client model is designed from the ground up with all aforementioned limitations in mind. As shown in Figure 2, the client model implements the same training pipeline as the source model in order to ensure maximum parameter reuse. After applying the pre-trained weights to the feature extractor f_i, the adaptation module A_i, and the classifier g_i, the parameter sets of the feature extractor and the classifier are frozen, resulting in these components being fixed during training. This strategy is beneficial in many ways: disabling the backpropagation of error especially through the deep feature extractor, thus avoiding computationally expensive gradient calculations of the corresponding weights, lowers the hardware requirements for the client model significantly. Furthermore, the network training is optimized to achieve a convergent solution at an accelerated pace while maintaining stability. Lastly, exclusive fine-tuning of the parameters of the adaptation module contributes to the concept of keeping domain-specific information in a single, purpose-built model component.

The adaptation of client i is conducted using a reduced k-shot target support set $D_{T_i}^{spt} \subset D_{T_i}$, where $k = \{3,5,10\}$ denotes the number annotated samples per class, reflecting data scarcity in the target domain. We further discuss and evaluate the selection of k in Section 4. The training objective for the fine-tuning task of client i is defined as

$$\underset{\psi_i}{\arg\min}\, \mathcal{L}_{CE}(\mathcal{F}(\psi_i;\tilde{x}_l),y_l) \tag{3}$$

with $\mathcal{F}(\cdot)$ being the client decision function parameterized with ψ_i, $\tilde{x}_l$ the augmented data sample drawn from the target domain data set $D_{T_i}^{spt}$, and the corresponding ground-truth label y_l, respectively.

In the next step (see Figure 2), target prototypes are calculated in the same manner as class prototypes in ProtoNet (Snell et al., 2017) and FedProto (Tan et al., 2022), but in *FedAcross* with a particular fine-tuning strategy. Our choice of the prototypical representation is based on its high interpretability, simplicity of computation, and memory efficiency. Therefore, the prototype to model the n-th class on client i is denoted as:

$$\omega_i^{(n)} = \frac{1}{|D_{T_i}^{spt},n|}\sum_{(x_l,y_l)\in D_{T_i}^{spt}} \tau_i(\tilde{x}_l) \tag{4}$$

where $\tau_i(\cdot)$ defines the embedding function over the client-side feature extractor and adaptation module with $\tau_i(\tilde{x}_l) = \mathcal{A}_i(f_i(\tilde{x}_l))$. The output set of the client adaptation is a collection of prototypes tailored to the target data set.

3.4 Client Inference

Inference on a client device is straightforward: The entire embedding pipeline, including feature extractor f_i and adaptation module $\mathcal{A}_i$, is upcycled to project the unlabeled, transformed sample $\tilde{x}_l$, observed on client device i, to generate the corresponding query prototype as illustrated in Figure 3[3]. The embedded query

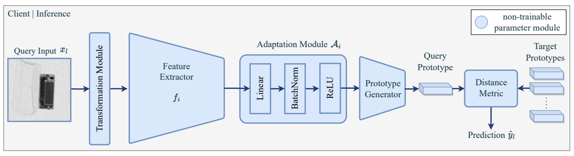

Figure 3: Client Inference .

vector is then fed into the *Distance Metric Module* for computation of the pairwise L2 distance between the query vector and the pre-computed target prototypes. The query sample is assigned to the class belonging to the nearest target prototype denoted as

$$\hat{y}_l = \arg\min_n \left\| \tau_i(\tilde{x}_l) - \omega_i^{(n)} \right\|_2 \tag{5}$$

where $\hat{y}_l$ is the predicted label of sample $\tilde{x}_l$ observed on client i.

3.5 Client Prototype Upstreaming

Clients can not only benefit from the FL cycle, but also pledge to contribute to it through sharing their locally refined knowledge without risking the exposure of sensitive information. We argue that access to raw client data points is restraint in three ways: First, in contrast to traditional FL our approach does not rely on interchanging model gradients, thus avoiding the threat of input data reconstruction from intercepted model gradients. Second, target prototypes reside at the mean of their respective class in the embedding space, restricted to only leak information in the same way that mean value statistics leak information (Brinkrolf et al., 2019). Third, even in case an adversary manages to reconstruct the feature vector of a single data point and additionally gains access to the fine-tuned client model, the resembling of a raw client data point encoded by a deep backbone is considered to be not a practically feasible task.

Prototype Upstreaming enables client devices to send their generated target prototypes and adaptation module parameters back to the server, minimizing data transfer whilst addressing bandwidth constraints and transmission latency. The prototypes are a

```
Input: $D_S, \phi_S, \psi_S, \nu_S \quad D^{spt}_{T_i}, \psi_i, i = 1, ..., I$

Part 1 – Server Model Training
1:  Initialize server model.
2:  for all pre-train epoch do
3:      for all batch $(x_S, y_S) \in D_S$ do
4:          Transform input $x_S$ by $\mathcal{T}$ and compute loss by Eq. 2.
5:          Update sever model parameters according to the loss.
6:      end for
7:  end for

Part 2 – Client Adaptation
1:  for all new client $i$ do
2:      Initialize client model with $\phi_i = \phi_S, \psi_i = \psi_S, \nu_i = \nu_S$.
3:      Freeze parameters $\phi_i$ and $\nu_i$.
4:      for all local epoch do
5:          for all batch $(x_l, y_l) \in D^{spt}_{T_i}$ do
6:              Transform input $x_l$ by $\mathcal{T}$and compute loss by Eq. 3.
7:              Update parameters $\psi_i$ according to the loss.
8:          end for
9:      end for
10:     Create client prototypes by Eq. 4.
11:     if UPSTREAM then
12:         Upload client prototypes $\omega_i$ and parameters $\psi_i$.
13:     end if
14: end for

Part 3 – Client Inference
1:  Compute label $\hat{y}_l$ by prototype inference Eq. 5.
```

Algorithm 1: *FedAcross*.

highly compact injective encoding of the former training data. Similarly to how clients generate target prototypes, the server can produce source prototypes by applying Equation (4) on the source data set D_S after pre-training the source model. Compared with target prototypes, these prototypes are more robust to outliers, yet they are also more specialized to the source data set. To compensate for this shortcoming, there are a variety of methods to enrich source prototypes with fine-tuned prototypes received from client devices, e.g. by applying an appropriate fusion strategy as described in (Tan et al., 2022). For enhanced bootstrapping of new clients in the FL cycle, the initial weights of the adaptation module $\mathcal{A}$ can be refined by processing the fine-tuned adaptation parameters received from previous clients. FedAvg (McMahan et al., 2017) is one of the best known approaches to combine model parameters within a FL context, where weights are collected from remote devices and averaged on a central hub. This method can be flawlessly integrated into our client-server setup. The pseudo-code for *FedAcross* is given in Algorithm 1.

[3]We show one of the X-ray images from the WeSort.AI waste detection scenario. The rechargeable battery of a cell phone has to be detected.

4 EXPERIMENTS

In this section, we explain the experiments of our approach replicating a garbage classification scenario[4].

4.1 Implementation Setup

All experiments are conducted to reflect the real-world challenges induces by domain shift over client observations, shortage of annotations on target samples and additional restrictions imposed by the FL environment. For each experiment, the feature extractor of the server instance is first initialized with a ResNet-34 or ResNet-50 architecture using corresponding weights from pre-training on ImageNet (Russakovsky et al., 2015), respectively. Additionally, the weight parameters of the adaptation module and the classifier's linear layers are initialized with random values from a normal distribution. The batch normalization layer's weights, on the other hand, are initialized using the Xavier normal initialization method (Glorot and Bengio, 2010). The main objective of the *server pre-training* is the optimization of the server model to recognize source domain-specific classes by minimizing the cross-entropy loss from Equation (2) on the full source data set D_S. Following (Guo et al., 2020) for a pre-training configuration in conjunction with few-shot learning downstream tasks, an SGD optimizer with initial learning rate of 0.01, momentum of 0.9 and weight decay of 0.001 is deployed. The learning rate is steadily reduced by learning rate scheduling. Server training runs 300 epochs, processing randomly shuffled mini-batches of size 128 per epoch, with optional early stopping on convergence. The server-side transformation module follows the recommendation from (Kim et al., 2022) by applying horizontal flipping, random resized cropping and color jittering to augment the input data. Following the pre-training step, the Flower-based server opens a gRPC network connection and listens for clients to join the FL round.

Fine-tuning starts by booting up client instances signaling their availability to the server with a pre-configured parameter set taken from the pre-training stage. Moreover, to further replicate the scarcity of labeled target data, each client i is restricted to only access its corresponding k-shot support set $D_{T_i}^{spt}$ during training, where each support set is randomly generated from the respective domain of the DA data set under inspection. The objective of the client training is to fine-tune the client model by minimizing the cross-entropy loss from Equation (3). The training setup for clients and server is equal, except that the client has a learning rate set to 0.1 and the mini-batch size is set to 32. For simulation purposes, the number of training epochs is set to 200 for one federated round since clients can access all support samples straight away. In real world scenarios, the training epochs can be split and distributed over the number of federated rounds. On completion of the fine-tuning process, the client creates the target prototypes based on Equation (4). Test accuracy is reported by evaluating each client individually using mean-centroid classification on its respective hold-back test set and average classification accuracy over five runs.

[4]The original x-ray and multi-spectral image data could not be made available for copyright reasons, but the data in the experiments are sufficiently similar.

4.2 Model Evaluation

In order to adequately benchmark our model, we first evaluate our method against approaches of source-free unsupervised DA as set up in (Zhang et al., 2022), focusing on single-domain performances. We chose the official Office-31 (Saenko et al., 2010) and OfficeHome (Venkateswara et al., 2017) data sets to be the most suitable fit for evaluation purposes, since the contained domains are based on pictures taken from real-world objects with visual differences in terms of lighting conditions, viewpoints and backgrounds. A total of 31 object classes with 4110 images are present in the Office-31 data set spread over three domains: Amazon (**A**), DSLR (**D**) and Webcam (**W**). The OfficeHome data set used in the second experiment includes 15500 images and 65 object classes divided into four domains: Art (**A**), Clipart (**C**), Product (**P**) and Real World (**RW**). For both experiments, we initially select a source domain (e.g. $D_S = \mathbf{A}$ sets domain **A** as source), pre-train the server model on it and copy the model to fine-tune it on the remaining domains, with $\mathbf{A} \rightarrow \mathbf{W}$ and $\mathbf{A} \rightarrow \mathbf{D}$ exemplifiying the mean-centroid classification task on the test set of the target domains **W** and **D**, respectively. The performance of *FedAcross* is evaluated using a ResNet-50 feature extractor, since all competitive methods utilize the latter. First, we compare our approach with source-free DA methods that permit access to the *full target* data set for fine-tuning: SHOT (Liang et al., 2020), SFDA (Kim et al., 2021) and SDAA (Kurmi et al., 2021). We also compare our approach to recent state-of-the-art few-shot adaptation methods FLUTE (Triantafillou et al., 2021), which develops a universal template based on multiple source data sets, and LCCS (Zhang et al., 2022) that adapts batch normalization statistics on target samples.

The experimental outcome on Office31 (Table 2) shows that *FedAcross* delivers on par adaptation re-

Table 2: Results[5] with ResNet-50 baseline, centralized source-free DA and few-shot transfer learning methods on Office-31. ”→” indicates a domain change.

Method	k	Office-31						
		$D_S = \mathbf{A}$		$D_S = \mathbf{W}$		$D_S = \mathbf{D}$		
		A → W	**A → D**	**W → A**	**W → D**	**D → A**	**D → W**	**Avg**
Baseline	-	68.4	68.9	60.7	99.3	62.5	96.7	76.1
SHOT	all	90.1	94.0	74.3	**99.9**	74.7	98.4	88.6
SFDA	all	91.1	92.2	71.2	99.5	71.0	98.2	87.2
SDDA	all	82.5	85.3	67.7	99.8	66.4	**99.0**	83.5
FLUTE*	5	84.6	88.2	66.4	99.1	66.4	95.3	83.3
LCCS*	5	92.8	91.8	**75.1**	99.9	**75.4**	98.5	88.9
FedAcross	5	89.4	90.4	63.5	94.5	60.0	90.4	81.4
FedAcross	10	**97.4**	**98.5**	71.1	98.6	71.0	97.4	**89.0**

sults against all competitors despite the more challenging conditions induced by the FL setup: Although LCCS produces the best overall adaptation performance (88.9%) with five labeled samples per class, *FedAcross* achieves the best overall adaptation results of all methods under inspection with $k = 10$ (89.0%). Ultimately, our approach offers practical advantages over LCCS for cross-device FL scenarios: First, *FedAcross* demonstrates enhanced flexibility as client adaptation does not depend on the number of batch normalization layers of the feature extractor, making it more versatile and applicable to a wider range of network architectures. Second, in contrast to *FedAcross*, the LCCS method requires a two-stage adaptation process, starting with a compute-intensive grid search in the initial stage. This demanding computational task is dedicated to determine the optimal parameter configuration for its learnable coefficients, which are then applied to kick-start the gradient update stage.

The results on the more challenging OfficeHome benchmark data set in Table 3 reveal that the unsupervised DA method SHOT outperforms all other competitors in that scenario, underlining the difficulty of adaptation in low data regimes (71.8% SHOT - 70.9% *FedAcross*, $k = 10$). We argue against SHOT that it requires on average six times the amount of (unlabeled) data points per class in the OfficeHome setup (59.6 images/class for SHOT - k images/class for *FedAcross*, $k = 10$) to achieve only slightly better overall accuracy than *FedAcross*.

Two further insights regarding our problem emerge from the results: There is a trade-off between the number of parameters that needs to be transmitted to the client initially (communication efforts) and on-client adaptation performance determined by the selection of the feature extractor. Moreover, the number of ground truth annotations k, is essential in enhancing prediction accuracy according to the specific needs of client operators.

To investigate the effectiveness of our approach in terms of *waste item classification*, the DA benchmark

Table 3: Results[5] with ResNet-50 baseline, centralized source-free DA and few-shot transfer learning methods on OfficeHome. ”→” indicates a domain change.

Method	k	OfficeHome												
		$D_S = \mathbf{A}$			$D_S = \mathbf{C}$			$D_S = \mathbf{P}$			$D_S = \mathbf{RW}$			
		A → C	**A → P**	**A → RW**	**C → A**	**C → P**	**C → RW**	**P → A**	**P → C**	**P → RW**	**RW → A**	**RW → C**	**RW → P**	**Avg**
Baseline	-	34.9	50.0	58.0	37.4	41.9	46.2	38.5	31.2	60.4	53.9	41.2	59.9	46.1
SHOT	all	57.1	**78.1**	**81.5**	68.0	**78.2**	**78.1**	67.4	54.9	82.2	73.3	58.8	**84.3**	**71.8**
SFDA	all	48.4	73.4	76.9	64.3	69.8	71.7	62.7	45.3	76.6	69.8	50.5	79.0	65.7
FLUTE*	5	49.0	70.1	68.2	53.8	69.3	65.1	53.2	46.8	70.8	**59.4**	51.7	77.3	61.2
LCCS*	5	**57.6**	74.5	77.0	60.0	71.5	70.9	59.2	54.7	75.9	69.2	**61.2**	81.5	67.8
FedAcross	5	45.9	68.9	66.1	53.9	67.6	64.8	55.8	47.2	67.2	59.5	48.1	73.3	59.9
FedAcross	10	56.7	77.0	76.3	**69.1**	76.7	74.6	**69.5**	59.4	76.6	72.4	60.4	81.5	70.9

data sets OfficeHome and DomainNet (Peng et al., 2019) (30 waste object classes, Clipart and Real domain) are modified to only include items typically observed in waste sorting scenarios[6]. In our experiment, the waste sorting service provider (**Srv**) pretrains its source model on all available waste object classes. Waste sorting facilities (**Cl**) fine-tune their local model on a specialized, randomly selected subset of ten classes, respectively. In Table 4, the prediction accuracy averaged over five runs with $k = \{0, 3, 5, 10\}$ is reported. The results for OfficeHome (Waste) show that *FedAcross* effectively improves the prediction accuracy on client devices with $k > 3$ in a photographic adaptation task, scaling with an increased number of annotated samples. The more challenging DomainNet (Waste) adaptation tasks across two domains with a larger distributional gap show similar performance improvements, thus highlighting the flexibility of *FedAcross*.

Table 4: Adaptability of *FedAcross* in a waste sorting scenario. ”→” indicates a domain change.

	OfficeHome (Waste)		DomainNet (Waste)	
k	$Srv_{\mathbf{RW}} \rightarrow Cl_{\mathbf{P}}$	$Srv_{\mathbf{P}} \rightarrow Cl_{\mathbf{RW}}$	$Srv_{\mathbf{C}} \rightarrow Cl_{\mathbf{R}}$	$Srv_{\mathbf{R}} \rightarrow Cl_{\mathbf{C}}$
0	87.82±0.26	78.68±0.86	54.51±0.12	65.0±0.50
3	84.42±0.25	75.73±0.73	54.74±0.06	69.0±0.65
5	89.43±0.45	84.44±0.29	57.77±0.24	76.87±0.25
10	93.45±0.56	88.91±0.19	66.48±0.32	83.18±0.53

Finally, we take advantage of the interpretable nature of our approach to visualize the separation progress over multiple adaptation stages using t-SNE (van der Maaten and Hinton, 2008) on the Office-31 classification task $\mathbf{A} \rightarrow \mathbf{W}$. One objective of our approach is to determine the optimal projection that will bring samples from the same class closer together while pushing samples from different classes further apart, resulting in prototypes with greater representativeness. In Figure 4, the plots illustrate target sample feature projections of five classes using: *(a)* an off-the-shelf ResNet-50 backbone, *(b)* a model pre-trained on D_S and *(c)* a model pre-trained on D_S and fine-tuned on $D_{T_i}^{spt}$ with their prototypes denoted as red rectangles, respectively.

[5]*Results referenced from (Zhang et al., 2022)

[6]Waste object classes are specified in the *FedAcross* sources.

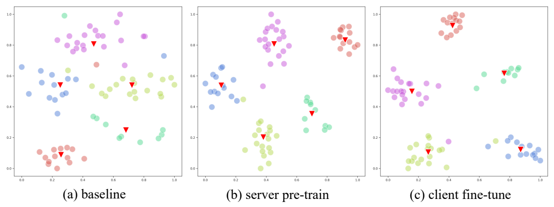

Figure 4: t-SNE plot of target class data (color-coded dots) and respective target class prototype (red triangles).

5 CONCLUSION

Although baseline prototypes are relatively close to each other, the pre-training assists in the initial class sample separation. Fine-tuning compresses samples of the same class even further, creating a suitable basis to apply a nearest-centroid classifier. In this work, we presented *FedAcross*, a computation-efficient FL approach offering a ready-to-deploy solution for target adaptation tasks under resource restrictions and distributional shifts across data silos. We demonstrated the scalability and flexibility of our method by exemplifying an image recognition task motivated from intelligent waste sorting systems throughout this paper. By employing prototype-based few-shot learning in combination with cross-device domain adaptation techniques, our model achieves competitive results in a federated server-client environment whilst keeping communication and computation efforts to a minimum. An extensive set of experiments performed on both public and industry data sets have demonstrated the applicability of our proposed approach in production environments.

While our current approach offers significant insights, it also opens up several avenues for *future research*. An immediate extension of our work could involve adapting our methodology to handle data streams in a federated learning environment. This evolution would require developing robust techniques to manage the dynamic and potentially large-scale nature of streaming data. Furthermore, integrating active learning strategies into federated clients presents an exciting opportunity. Such an approach would not only address the challenge of expansive labeling in distributed settings but also enhance the efficiency of the learning process. A critical aspect of this future work would be the quality assessment of data points obtained from streaming data, ensuring that the most informative samples contribute to the learning process. This progression would significantly improve the model's adaptability and performance in real-world, dynamic scenarios. Additionally, exploring the impact of these advancements on privacy preservation and communication efficiency in federated settings could provide valuable insights, aligning with the growing need for secure and scalable machine learning solutions. Ultimately, these efforts would contribute to the development of more sophisticated, efficient, and practical federated learning systems, capable of handling the complexities of real-world data distributions and applications.

ACKNOWLEDGEMENTS

MR and MM thank the Bavarian HighTech Agenda and the Würzburg Center for Artificial Intelligence and Robotics (CAIRO).

REFERENCES

Bashkirova, D., Mishra, S., Lteif, D., Teterwak, P., Kim, D., Alladkani, F. M., Akl, J., Çalli, B., Bargal, S. A., Saenko, K., Kim, D., Seo, M., Jeon, Y., Choi, D.-G., Ettedgui, S., Giryes, R., Hussein, S. A., Xie, B., and Li, S. (2023). VisDA 2022 challenge: Domain adaptation for industrial waste sorting. *CoRR*, abs/2303.14828.

Beutel, D. J., Topal, T., Mathur, A., Qiu, X., Fernandez-Marques, J., Gao, Y., Sani, L., Kwing, H. L., Parcollet, T., Gusmão, P. P. d., and Lane, N. D. (2020). Flower: A friendly federated learning research framework.

Brinkrolf, J., Göpfert, C., and Hammer, B. (2019). Differential privacy for learning vector quantization. *Neurocomputing*, 342:125–136.

Chang, W.-G., You, T., Seo, S., Kwak, S., and Han, B. (2019). Domain-specific batch normalization for unsupervised domain adaptation. *CoRR*, abs/1906.03950:7346–7354.

Dhillon, G. S., Chaudhari, P., Ravichandran, A., and Soatto, S. (2020). A baseline for few-shot image classification. In *International conference on learning representations*, volume abs/1909.02729 of *International Conference on Learning Representations*. OpenReview.net.

Falcon, W. and team, T. P. L. (2019). PyTorch lightning.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In Teh, Y. W. and Titterington, M., editors, *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, volume 9 of *Proceedings of machine learning research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy. PMLR.

Guo, Y., Codella, N. C., Karlinsky, L., Codella, J. V., Smith, J. R., Saenko, K., Rosing, T., and Feris, R. (2020). A broader study of cross-domain few-shot learning. In Vedaldi, A., Bischof, H., Brox, T., and Frahm, J.-M., editors, *Computer vision – ECCV 2020*, volume 12372 of *European Conference on Computer Vision*, pages 124–141. ECCV.

Hard, A., Rao, K., Mathews, R., Beaufays, F., Augenstein, S., Eichner, H., Kiddon, C., and Ramage, D. (2018). Federated learning for mobile keyboard prediction. *ArXiv*, abs/1811.03604.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE conference on computer vision and pattern recognition (CVPR)*, Computer Vision and Pattern Recognition, pages 770–778. IEEE.

Hu, S. X., Li, D., Stühmer, J., Kim, M., and Hospedales, T. M. (2022). Pushing the limits of simple pipelines for few-shot learning: External data and fine-tuning make a difference. In *IEEE/CVF conference on computer vision and pattern recognition, CVPR 2022, new orleans, LA, USA, june 18-24, 2022*, Computer Vision and Pattern Recognition, pages 9058–9067. IEEE.

Huang, W., Ye, M., Shi, Z., Li, H., and Du, B. (2023). Rethinking federated learning with domain shift: A prototype view. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16312–16322, Los Alamitos, CA, USA. IEEE Computer Society.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Nitin Bhagoji, A., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., D'Oliveira, R. G. L., Eichner, H., El Rouayheb, S., Evans, D., Gardner, J., Garrett, Z., Gascón, A., Ghazi, B., Gibbons, P. B., Gruteser, M., Harchaoui, Z., He, C., He, L., Huo, Z., Hutchinson, B., Hsu, J., Jaggi, M., Javidi, T., Joshi, G., Khodak, M., Konecný, J., Korolova, A., Koushanfar, F., Koyejo, S., Lepoint, T., Liu, Y., Mittal, P., Mohri, M., Nock, R., Özgür, A., Pagh, R., Qi, H., Ramage, D., Raskar, R., Raykova, M., Song, D., Song, W., Stich, S. U., Sun, Z., Suresh, A. T., Tramèr, F., Vepakomma, P., Wang, J., Xiong, L., Xu, Z., Yang, Q., Yu, F. X., Yu, H., and Zhao, S. (2021). Advances and open problems in federated learning. *Found. Trends Mach. Learn.*, 14(1–2):1–210.

Kim, Y., Cho, D., Han, K., Panda, P., and Hong, S. (2021). Domain adaptation without source data. *IEEE Transactions on Artificial Intelligence*, 2(6):508–518.

Kim, Y., Oh, J., Kim, S., and Yun, S.-Y. (2022). How to fine-tune models with few samples: Update, data augmentation, and test-time augmentation.

Koh, P. W., Sagawa, S., Marklund, H., Xie, S. M., Zhang, M., Balsubramani, A., Hu, W., Yasunaga, M., Phillips, R. L., Gao, I., Lee, T., David, E., Stavness, I., Guo, W., Earnshaw, B. A., Haque, I. S., Beery, S., Leskovec, J., Kundaje, A., Pierson, E., Levine, S., Finn, C., and Liang, P. (2021). WILDS: A benchmark of in-the-Wild distribution shifts. In Meila, M. and 0001, T. Z., editors, *International conference on machine learning (ICML)*, volume 139 of *International Conference on Machine Learning*, pages 5637–5664. PMLR.

Kurmi, V. K., Subramanian, V. K., and Namboodiri, V. P. (2021). Domain impression: A source data free domain adaptation method. *CoRR*, abs/2102.09003:615–625.

Laier, N. and Laier, J. (2023). WeSort.AI homepage. https://www.wesort.ai/. Accessed: 2023-10-24.

Lange, J.-P. (2021). Managing plastic waste-sorting, recycling, disposal, and product redesign. *ACS Sustainable Chemistry & Engineering*, 9(47):15722–15738.

Li, W., Liu, X., and Bilen, H. (2022). Cross-domain few-shot learning with task-specific adapters. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, Computer Vision and Pattern Recognition, pages 7151–7160, Los Alamitos, CA, USA. IEEE Computer Society.

Liang, J., Hu, D., and Feng, J. (2020). Do we really need to access the source data? Source hypothesis transfer for unsupervised domain adaptation. In *Proceedings of the 37th international conference on machine learning*, ICML'20, pages 6028–6039. JMLR.org.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and Arcas, B. A. y. (2017). Communication-efficient learning of deep networks from decentralized data. In Singh, A. and Zhu, J., editors, *Proceedings of the 20th international conference on artificial intelligence and statistics*, volume 54 of *Proceedings of machine learning research*, pages 1273–1282.

Müller, R., Kornblith, S., and Hinton, G. (2019). When does label smoothing help? In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché Buc, F., Fox, E. A., and Garnett, R., editors, *Neural Information Processing Systems*, pages 4696–4705. Curran Associates Inc., Red Hook, NY, USA.

Nado, Z., Padhy, S., Sculley, D., D'Amour, A., Lakshminarayanan, B., and Snoek, J. (2021). Evaluating prediction-time batch normalization for robustness under covariate shift.

Peng, X., Bai, Q., Xia, X., Huang, Z., Saenko, K., and Wang, B. (2019). Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE international conference on computer vision*, IEEE International Conference on Computer Vision, pages 1406–1415. IEEE.

Perez, L. and Wang, J. (2017). The effectiveness of data augmentation in image classification using deep learning.

Raab, C., Röder, M., and Schleif, F.-M. (2022). Domain adversarial tangent subspace alignment for explainable domain adaptation. *Neurocomputing*, 506:418–429.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

Saenko, K., Kulis, B., Fritz, M., and Darrell, T. (2010). Adapting visual category models to new domains. In Daniilidis, K., Maragos, P., and Paragios, N., editors, *Computer vision – ECCV 2010*, volume 6314 of *European Conference on Computer Vision*, pages 213–226, Berlin, Heidelberg. Springer Berlin Heidelberg.

Siqueira, F. and Davis, J. G. (2021). Service computing for industry 4.0: State of the art, challenges, and research opportunities. *Acm Computing Surveys*, 54(9):1–38.

Snell, J., Swersky, K., and Zemel, R. (2017). Prototypical networks for few-shot learning. In Guyon, I., Luxburg, U. v., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R., editors, *Advances in neural information processing systems*, NIPS, pages 4077–4087.

Song, Y., Wang, T., Cai, P., Mondal, S. K., and Sahoo, J. P. (2023). A comprehensive survey of few-shot learning: Evolution, applications, challenges, and opportunities. *Acm Computing Surveys*, abs/2205.06743.

Tan, Y., Long, G., Liu, L., Zhou, T., Lu, Q., Jiang, J., and Zhang, C. (2022). FedProto: Federated prototype learning across heterogeneous clients. In *AAAI conference on artificial intelligence*, volume 36 of *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8432–8440. Association for the Advancement of Artificial Intelligence (AAAI).

Triantafillou, E., Larochelle, H., Zemel, R., and Dumoulin, V. (2021). Learning a universal template for few-shot dataset generalization.

van der Maaten, L. and Hinton, G. (2008). Visualizing high-dimensional data using t-SNE. *Journal of Machine Learning Research*, 9(nov):2579–2605.

Venkateswara, H., Eusebio, J., Chakraborty, S., and Panchanathan, S. (2017). Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, Computer Vision and Pattern Recognition, pages 5018–5027. IEEE.

Wang, Y., Yao, Q., Kwok, J., and Ni, L. (2020). Generalizing from a few examples: A survey on few-shot learning. *ACM Computing Surveys*, 53:1–34.

Yang, K., Shi, Y., Zhou, Y., Yang, Z., Fu, L., and Chen, W. (2020). Federated machine learning for intelligent IoT via reconfigurable intelligent surface. *IEEE Network*, 34:16–22.

Yang, Q., Liu, Y., Chen, T., and Tong, Y. (2019). Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology*, 10(2):1–19.

Yang, T., Andrew, G., Eichner, H., Sun, H., Li, W., Kong, N., Ramage, D., and Beaufays, F. (2018). Applied federated learning: Improving google keyboard query suggestions.

Zhang, W., Shen, L., Zhang, W., and Foo, C.-S. (2022). Few-shot adaptation of pre-trained networks for domain shift. In Raedt, L. D., editor, *Proceedings of the thirty-first international joint conference on artificial intelligence, IJCAI-22*, volume abs/2205.15234 of *International Joint Conference on Artificial Intelligence*, pages 1665–1671. International Joint Conferences on Artificial Intelligence Organization.

Zhao, C., Sun, X., Yang, S., Ren, X., Zhao, P., and McCann, J. (2022). Exploration across small silos: Federated few-shot learning on network edge. *IEEE Network*, 36(1):159–165.