EFFEKT: Efficient Federated Knowledge Transfer to Foundation Models

Anonymous authors
Paper under double-blind review

Abstract

Recent data protection laws have accelerated the adoption of Federated Learning (FL) for privacy-preserving decentralized training. Nevertheless, increasing model sizes imposes substantial computational demands on client devices, limiting FL applicability in resource-constrained settings. We introduce a novel multi-domain federated learning framework in which lightweight client-side proxy models collaborate with a server-side Foundation Model (FM) to learn new concepts without sharing private data. Our approach, EFFEKT, enables efficient server-side training of domain-specific LoRA adapters while preserving feature-space alignment between the FM and proxy extractors via novel bi-directional cross-distillation strategies. Experiments on multiple real-world datasets and deployments on low-power edge devices demonstrate consistent improvements over state-of-the-art baselines while maintaining lightweight computation at client side.

1 Introduction

Federated Learning (FL) has emerged as a successful paradigm in machine learning, addressing the critical need for privacy-aware distributed training of models. This approach typically involves local optimization on client devices coupled with server-side aggregation, dissemination, and control (Shenaj et al., 2023b). The key advantage of FL lies in its ability to train models using locally available data on client devices, thereby preserving privacy by eliminating the need to transmit sensitive information to a central server. While FL has proven effective for training lightweight models on mobile devices, the recent advent of high-performing yet extremely complex transformer-based foundation models presents a significant challenge. These sophisticated models, due to their sheer size and computational requirements, are often too complex for deployment and training on client devices, potentially limiting the applicability of traditional FL approaches.

To illustrate the practical implications of this challenge, consider a scenario where a technology company aims to offer a personalized remote image recognition service exploiting a powerful foundation model at server side. The local implementations of this service could be tailored to specific domains (*e.g.*, plant or insect species recognition, or vehicle model identification) and adapted to individual user data (that could have strong privacy constraints), while the central server maintains a more comprehensive, multi-domain model. In this context, the ability to improve the global model using local, domain-specific data in a privacy-preserving manner becomes crucial. However, the complexity of state-of-the-art vision models often precludes their direct deployment on user devices, necessitating novel federated learning techniques.

A viable solution for this issue is to deploy a simple CNN model on client devices, with its feature space aligned to that of a larger, more complex model. This idea is exploited by FedPromo (Caligiuri et al., 2025), where the lightweight network served as a proxy for the local training of a task-specific decoder, which was then reattached to the main model. While this approach allows tackling the computational complexity issues, from a performance viewpoint, it is still hampered by the issues of the simple averaging of network weights used at the server side, especially when client models become too misaligned (McMahan et al., 2017).

In this paper, we introduce a novel server-side aggregation approach that allows for better performances by replacing standard weight averaging with a distillation-based scheme. Our method builds upon the concept of using a simple network with features aligned to the complex one. However, at each federated round, we

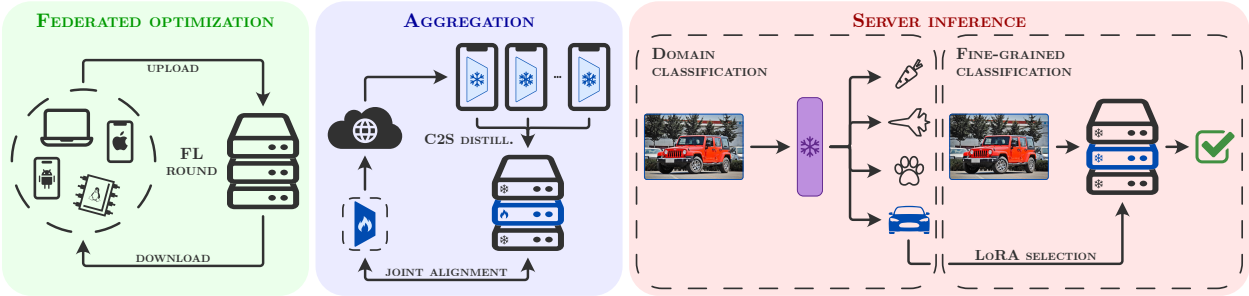

Figure 1: EFFEKT is a heterogeneous federated learning pipeline that allows bidirectional, multidomain optimization of a server-side foundation model and small client-side proxy models.

first efficiently distill the information learned by the clients into the foundation model, leveraging Low-Rank Approximation (LoRA). The updated foundation model is then utilized to perform joint re-alignment between the server and proxy models before initiating the new federated round.

Crucially, our new approach yields substantial improvements in server-side accuracy, with an average increase over the state-of-the-art of 3.9% for top-1 accuracy and 2.7% for top-5 accuracy across 5 fine-grained domains. To further validate our setup and its real-world applicability, we deploy the architecture on compute-constrained devices. We measure final accuracy, energy consumption, and bandwidth usage in these deployments. Our results demonstrate that the simulated experiments closely mirror real-world deployments, exhibiting limited energy consumption and network usage, thus enabling deployment on personal devices.

This advancement not only enhances the performance of federated learning systems but also expands their potential applications. In the context of our earlier example, it allows for more effective learning of new, previously unknown classes at the server level without direct access to client data. This capability is particularly valuable in scenarios where the central model must evolve to recognize novel categories (e.g., rare animal species or new vehicle models) based solely on distributed learning from user devices, all while maintaining strict privacy standards.

By bridging the gap between complex foundation models and resource-constrained edge devices, our approach paves the way for more inclusive and privacy-preserving AI applications, enabling a wider range of devices to participate in and benefit from advanced machine learning models.

2 Related Works

Federated Learning (FL) was introduced by McMahan et al. (2017), which proposed a simple server-side aggregation approach based on weight averaging. The approach, called FedAvg, was effective in simple scenarios but struggled when tackling strongly non-IID data distributions. The technique has been extended in many different directions (Yuan et al., 2024; Shenaj et al., 2023b). For example, FedProx Li et al. (2020) added a proximal term to the local objectives to limit the impact of local updates to reduce clients' drift. Similarly, SCAFFOLD (Karimireddy et al., 2020) tackles client drift through direct estimation and subsequent correction. MOON (Li et al., 2021) exploited model-based contrastive learning to improve client training.

Foundation Models (FMs) enabled significant improvements in several fields, from Natural Language Processing (*e.g.*, BERT (Devlin et al., 2019)) to Computer Vision. The CLIP (Radford et al., 2021) approach represents a key advancement in Vision-Language Models (VLMs); by using contrastive learning, it enables the alignment of visual and text representations. Another cornerstone work is DINO (Caron et al., 2021), which learns the semantics of objects without supervision by utilizing self-distillation. FMs have also been used as oracle feature extractors for pretraining of other network models (Ostapenko et al., 2022).

Pretraining in FL: In the standard FL settings (McMahan et al., 2017), optimization begins with randomly initialized model weights; however, this can affect efficiency and performance, particularly in cases of high heterogeneity. To tackle this issue, server-side pretraining strategies have been employed to stabilize subsequent

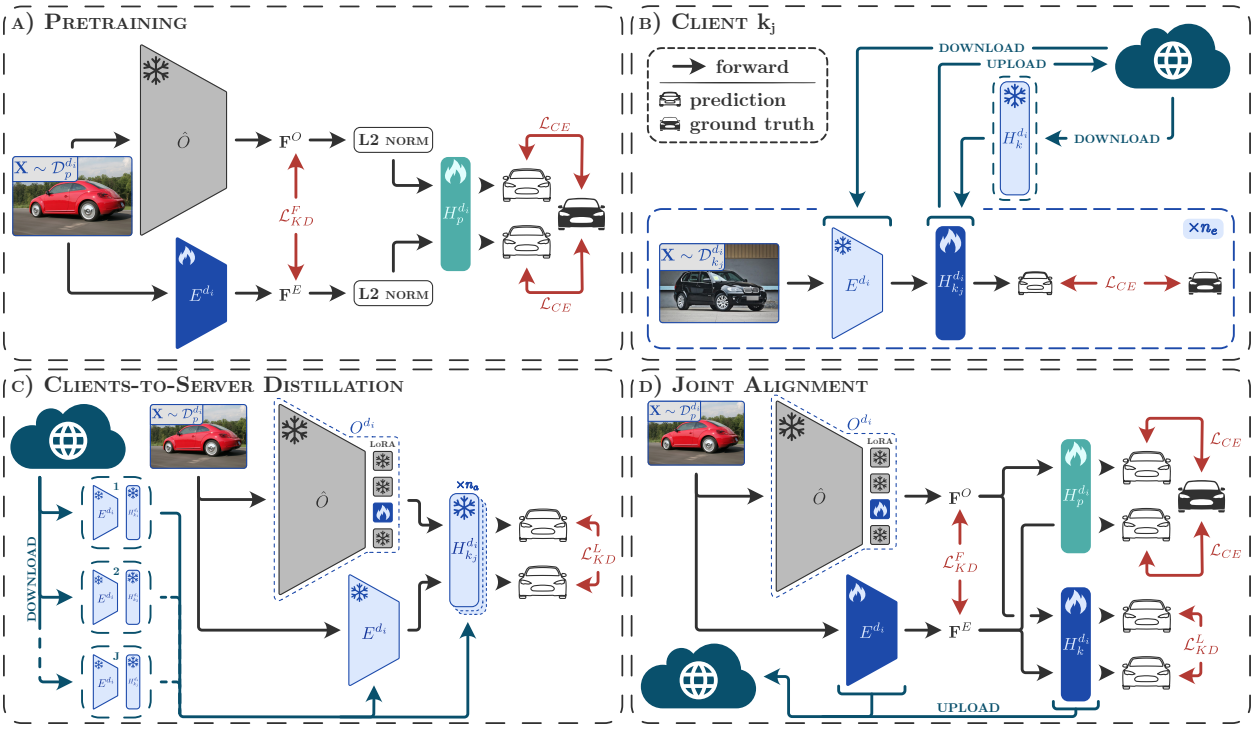

Figure 2: The EFFEKT architecture and cross-distillation strategies. a) pretraining pipeline; b) local update of client k_j. c) Clients-to-Server distillation (C2S), where the active clients' proxy models are used to update the correct domain-LoRA on the server; d) Joint Alignment (JA) distillation, utilizing the updated server model as initialization for the next round's proxy models.

federated training, enabling longer local training and reducing communication costs (Shenaj et al., 2023b; Nguyen et al., 2023; Chen et al., 2023). Pretraining on synthetically generated data has also been utilized to establish a stable starting point for challenging vision applications (Shenaj et al., 2023a).

Knowledge Distillation (KD): Knowledge Distillation is a widely used strategy that facilitates the transfer of knowledge across different learning models (Hinton, 2015). It has recently been exploited in FL, both in a Model-agnostic way (Jeong et al., 2018; Li & Wang, 2019; Hongyan et al., 2019; Wu et al., 2024) to improve FL across various architectures and in a data-agnostic way (Zhu et al., 2021; Sattler et al., 2021; Zhang et al., 2022; Yao et al., 2023) to enhance performance in non-IID settings. Inspired by these works, our approach uses KD during pretraining to align the lightweight model with the server model.

Federated Training of FMs has recently emerged as a novel venue of research (Ren et al., 2025), focusing on aggregation techniques suitable for large-scale models and on improving both computational and communication efficiency. Some works concentrate on the federated training of specific FMs: a federated training strategy for CLIP (Li et al., 2021) is proposed in Lu et al. (2023); the work of Harasic et al. (2024) analyzes the performance of DINO (Oquab et al., 2023) in federated settings; a pretrained SAM (Kirillov et al., 2023) is fine-tuned on the clients in Liu et al. (2024). However, training these models is expensive; thus, other approaches tackle FM optimization by training LoRA adapters in a federated manner instead of the full models (Yi et al., 2023; Babakniya et al., 2023). Note that these methods train the adapter on the client and average the weights (*e.g.*, via FedAvg) on the server. Finally, Caligiuri et al. (2025) exploits a lightweight client architecture whose latent space aligns with that of a large model at the server, allowing federated training without bringing the FM to the clients. Iterating on these works, our approach trains a FM in a scalable, modular, and privacy-preserving manner, using FL on remote devices and KD at server side.

3 Problem Setting

In this section, we briefly recap the Cross-Architecture Federated Knowledge Transfer (CA-FKT) task (Caligiuri et al., 2025) and the mathematical notation.

Following traditional federated learning, we assume the presence of a server and a set of n_k clients $\mathcal{K} = \{k_j\}_{j=1}^{n_k}$, which collaborate in a distributed optimization setup. Training is performed over n_r communication rounds. In each round, the server selects a different set of n_a active clients $\mathcal{K}_a$. Each client $k_j \in \mathcal{K}_a$ trains its model $M_{k_j} = H_{k_j} \circ E$ for n_e local epochs using its private dataset $\mathcal{D}_{k_j}$ before sending the updated weights to the server for aggregation (note that only H_{k_j} is trained). Then, the server samples a new client set $\mathcal{K}_a$ and sends to the clients the aggregated model as initialization for the next distributed optimization round. The setup introduces key differences from the standard approach.

Firstly, it supports domain-asynchronous training; therefore, the clients are divided into n_i non-overlapping groups $\mathcal{K}^{d_i}$ based on their domain index $i \in \mathcal{I} = \{1, \dots, n_i\}$. Since different domains require distinct sets of model parameters, for clarity of notation, we will add the domain index to the general notation introduced above for the rest of the document, *e.g.*, client k_j's model M_{k_j} becomes $M_{k_j}^{d_i} = H_{k_j}^{d_i} \circ E^{d_i}$, where $H_{k_j}^{d_i} : \mathbb{R}^{n_f} \mapsto \mathcal{Y}^{d_i} = \{1, \dots, n_c^{d_i}\}$ and $E^{d_i} : \mathbb{R}^{h \times w \times 3} \mapsto \mathbb{R}^{n_f}$ are the domain i head and encoder, respectively.

Secondly, we assume that the server and client model architectures are distinct; more specifically, the server encoder $\hat{O}$ is assumed to be much larger and better-performing than the small, efficient client encoders E^{d_i}. Crucially, to allow cross-distillation, the features produced by the two models must have the same dimensions, allowing us to attach the same head H^{d_i} to both the server and client encoders. Refer to Sec. 4 for more details. While $\hat{O}$ is assumed to work well across all domains, to improve performance, we introduce a set of LoRA (Hu et al., 2022) adapters $\{L^{d_i}\}_{i \in \mathcal{I}}$. For ease of notation, we define O^{d_i} as the server model adapted with LoRA L^{d_i}. Formally, given an input RGB image $\mathbf{X} \in \mathcal{X} \subset \mathbb{R}^{h \times w \times 3}$ with a resolution of $w \times h$ pixels, the server and client encoders extract features $\mathbf{F}^O = O^{d_i}(\mathbf{X})$ and $\mathbf{F}^E = E^{d_i}(\mathbf{X})$, where $\mathbf{F}^O, \mathbf{F}^E \in \mathbb{R}^{n_f}$.

In our setup, during training, the domain-specific clients notify the server of the correct domain index i; while during inference, the domain of a sample $\mathbf{X}$ can be inferred using a domain discriminator $D : \mathbb{R}^{h \times w \times 3} \mapsto \mathcal{I}$ trained on a set of public, task-specific, pretraining datasets $\{\mathcal{D}_p^{d_i}\}_{i \in \mathcal{I}}$ with different, coarser, class labels.

4 Proposed Method

In order to tackle the CA-FKT task, we implement our EFFEKT federated framework (summarized in Fig. 2) on top of an encoder-decoder classification network, where the architectures of the server and client encoders differ while maintaining a common latent space. This choice permits the deployment of larger and higher-performing models on the server while maintaining a low computational profile on the clients. While, in principle, any couple of feature-aligned architectures could be used, to ensure fair comparison, we choose the same architectures used in competing works (Caligiuri et al., 2025). More specifically, the server architecture is a DINOv2 (ViT-L/14-Reg) model (Oquab et al., 2023), while the client model is a MobilenetV3-small (Howard et al., 2019) with a linear feature translator module. The content of the two feature spaces is aligned during a pretraining phase that is different for each domain d_i, and uses an appropriate public task-specific dataset $\mathcal{D}_p^{d_i}$. This results in a specific classification head $H_p^{d_i}$ for each of the domains due to the different class sets; the head can be freely attached to both server and client encoders.

4.1 Federated Training

Our implementation of the CA-FKT task aims to closely mimic a real-world deployment, which incurs several issues during distributed optimization, chief among them client divergence due to the strongly non-IID class distribution (Hsu et al., 2019). Therefore, strong regularization objectives and effective FL techniques are necessary to control the evolution of training. As such, we decided to employ a selective weight optimization strategy for the distributed optimization framework, *i.e.*, the ICP approach (Caligiuri et al., 2025). More details are in Alg. A.2 of the Appendix. Simply using this strategy, however, leads to subpar performance at the server side; therefore, we propose a novel cross-distillation federated aggregation technique to update

Algorithm 1 Pseudocode for EFFEKT ServerRound.

Require: Number of rounds n_r and active clients n_a, domain index i, server encoder (with LoRA) O^{d_i}, peak learning rate $lr_{\max}$, cosine annealing learning rate scheduler, pretraining head $H_p^{d_i}$

Select the domain clients' group $\mathcal{K}^{d_i}$, and the initial model $M_k^{d_i} = H_k^{d_i} \circ E^{d_i}$
for $r \leftarrow 1 \dots n_r$ **do**
  Compute the active clients set $\mathcal{K}_a^{d_i}$ by randomly sampling n_a clients from $\mathcal{K}^{d_i}$ // Different each round
  Send to the active clients $j \in \mathcal{K}_a^{d_i}$ the model $M_k^{d_i}$
  Compute current round learning rate from the scheduler $lr \leftarrow \text{CosineScheduler}(lr_{\max}, r, n_r)$
  for $k_j \in \mathcal{K}_a^{d_i}$ **do** // Performed in Parallel
    Receive $H_{k_j}^{d_i} \leftarrow \text{ClientRound}\Big(H_k^{d_i}, lr\Big)$ from k_j
  end for
  $L^{d_i} \leftarrow \text{C2S}\Big(L^{d_i}, E^{d_i}, \{H_{k_j}^{d_i}\}_{k_j \in \mathcal{K}_a^{d_i}}\Big)$ // Train LoRAs
  $H_k^{d_i} \leftarrow \text{FedAvg}\Big(\{H_{k_j}^{d_i}\}_{k_j \in \mathcal{K}_a^{d_i}}\Big)$ // Aggregate heads
  $M_k^{d_i} \leftarrow H_k^{d_i} \circ E^{d_i}$ // Update model with new head
  $\Big(M_k^{d_i}, L^{d_i}, H_p^{d_i}\Big) \leftarrow \text{JA}\Big(M_k^{d_i}, L^{d_i}, H_p^{d_i}\Big)$ // Align
end for

the server model and improve its accuracy on unseen domains without impacting its generalizability using domain-specific LoRA adapters (Hu et al., 2022). More specifically, during each optimization round, each active client k_j receives the updated model from the server and trains only the head $H_{k_j}^{d_i}$ using its private local dataset $\mathcal{D}_{k_j}^{d_i}$, thus preserving latent space alignment between different clients. After local optimization, the updated heads are sent to the server for aggregation and FM tuning. As detailed in Alg. 1, this step is split into two complementary operations: *Clients-to-Server (C2S)* and *Joint Alignment (JA)* distillation.

Clients-to-Server (C2S) Distillation
As the name implies, the first step of our aggregation pipeline involves updating the server-side FM by leveraging the knowledge acquired by the clients during the current round. Note that, to reduce computational and storage costs, the update is applied only to a small set of LoRA parameters. This choice is a good tradeoff between accuracy and complexity as shown by the experiments in Tab. 7 of the ablation. The knowledge is updated using a logit-level knowledge distillation loss $\mathcal{L}_{KD}^{L}(\mathbf{P}, \mathbf{T}) = -\sum_{y \in \mathcal{Y}} \mathbf{P}[y] \log \mathbf{T}[y]$, where $\mathbf{P}$ is a vector of predicted class-probabilities and $\mathbf{T}$ is the reference distribution. Note that this is the reverse of the standard orientation, leading to several benefits in our setup. More specifically, this loss can be decomposed into $KL(P||T) + H(P)$, that is, the mode-covering (Hinton, 2015) version of Kullback-Leibler divergence (useful under unreliable supervision) and prediction entropy, which is often used as a loss in unsupervised domain adaptation (Springenberg, 2015). This objective is computed on the output probability of each received (frozen) client head and is summed to compute the final loss:

$$\mathcal{L}_{C2S} = \sum_{k_j \in \mathcal{K}_a} \mathcal{L}_{KD}^{L} \left(H_{k_j}^{d_i}(\mathbf{F}_O), H_{k_j}^{d_i}(\mathbf{F}_E) \right) \quad . \tag{1}$$

Note that, since the training is performed by the server that does not have access to private client data, the samples used for distillation are extracted from the task-specific pretraining dataset $\mathcal{D}_p^{d_i}$. A more detailed breakdown of this procedure is reported in Alg. A.3 of the Appendix.

Joint Alignment (JA) Distillation
In general, the updates applied to the server model in the previous step may lead to misalignment between the feature spaces of the FM and of the client-side proxies. Therefore, an additional realignment procedure is necessary to restore feature matching and improve the accuracy of the aggregated client model. Note that, in the previous step, the trained client heads were kept separate to preserve all available client knowledge. In contrast, here we begin by computing a single domain client head $H_k^{d_i}$ by aggregating the trained client heads $H_{k_j}^{d_i}, k_j \in \mathcal{K}_a^{d_i}$ using the FedAvg strategy (McMahan et al., 2017) to serve as the distillation target. This is necessary because a unique model must be sent to all active clients to initialize the subsequent federated

round. As before, to preserve the privacy of client data, this realignment is performed by the server using the task-specific pretraining data $\mathcal{D}_p^{d_i}$. The key differences lie in the optimization targets, which include both classification accuracy and alignment objectives. We train the task-specific head $H_p^{d_i}$, the full client model $M_k^{d_i}$ (including the encoder), and the LoRA L^{d_i}. The training loss has two major focuses. The first being the classification accuracy of both the adapted FM and the client encoder using standard cross-entropy objectives. The second is the preservation of feature compatibility, which is achieved by encouraging server and client latent representations to be similar using the $\mathcal{L}_{KD}^F$ loss together with bidirectional logits alignment $\mathcal{L}_{KD}^L$. The overall loss function for this step can be written as:

$$\begin{aligned}\mathcal{L}_{JA} = \mathcal{L}_{CE}\left(H_p^{d_i}(\mathbf{F}_O), y\right) + \mathcal{L}_{CE}\left(H_p^{d_i}(\mathbf{F}_E), y\right) + \mathcal{L}_{KD}^F\left(\mathbf{F}_E, \mathbf{F}_O\right) + \mathcal{L}_{KD}^F\left(\mathbf{F}_O, \mathbf{F}_E\right) \\ + \lambda_{KD}^L \mathcal{L}_{KD}^L\left(H_k^{d_i}(\mathbf{F}_O), H_k^{d_i}(\mathbf{F}_E)\right) + \lambda_{KD}^L \mathcal{L}_{KD}^L\left(H_k^{d_i}(\mathbf{F}_E), H_k^{d_i}(\mathbf{F}_O)\right) \quad ,\end{aligned} \tag{2}$$

where $\mathcal{L}_{CE}$ denotes the standard categorical cross-entropy loss (applied to $H_p^{d_i}$ to match the class-set of the only domain dataset available at server-side, $\mathcal{D}_p^{d_i}$), λ_{KD}^L is a hyperparameter, and $\mathcal{L}_{KD}^F$ is the sum of the L1, L2 and cosine distances (Barbato et al., 2024). The first two objectives are necessary to preserve the discriminative capability of the server and proxy encoders, respectively; the next two allow for shared application of the head; the last two update the aggregated head to ensure compatibility with the new latent space. The pseudocode is provided in Alg. A.4 of the Appendix.

4.2 Multi-Domain Setup

In our setup, each client knows its domain index and can communicate it to the server during training, thereby avoiding inter-domain confusion during aggregation. Note that in a realistic scenario, federated learning is often performed in a domain-asynchronous manner, meaning that the server must be able to activate the correct LoRA adapter and model head to optimize based on the received client updates. Our approach employs LoRAs and distinct heads instead of directly updating the server model, which guarantees the possibility of incrementally learning new domains without impacting the performance of previously learned models or compatibility with the compute-constrained proxy encoders.

Multi-Domain Inference
Whenever the server needs to run inference on a given sample, two scenarios are possible: either the prompt includes the expected domain, or the domain information is missing. In the former case, the server can directly activate the correct LoRA adapter and head before executing the model. In the latter case, the server must estimate the correct domain before proceeding. To tackle this issue, we introduce a server-side classifier that identifies the domain of the query.

Domain Discriminator
Since the domain discriminator may be utilized for each sample during server-side inference, we aimed to minimize its computational impact without compromising accuracy. Moreover, to avoid retraining the model every time a domain is added, we adopted a few-shot-based approach, *i.e.*, prototypical classification. More specifically, the domain classifier is implemented as a frozen encoder-only MobileNetV3-small (Howard et al., 2019) architecture pretrained on ImageNet-1k (Deng et al., 2009), which extracts a latent representation of each given query sample. At inference time, the extracted features are compared to a set of internally stored prototypes (one for each domain) using the L2-distance metric. The domain index i corresponding to the closest prototype is selected as output. The domain prototypes are computed during the pretraining step of the domain-specific client encoders using the task-specific pretraining datasets $\mathcal{D}_p^{d_i}$. Despite the negligible computational cost, this simple classifier achieves remarkable domain accuracy, as detailed in Table 3 of the ablation studies.

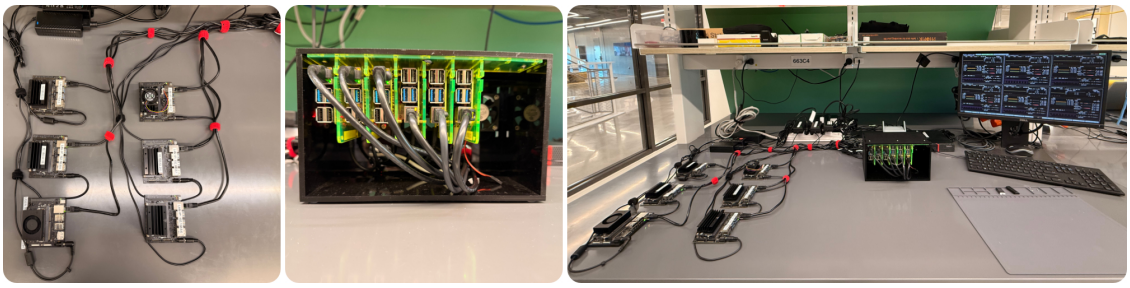

Figure 3: Real devices' setup: 3 Raspberry Pi 4, 3 Raspberry Pi 5, 5 Jetson Nano, and 1 Jetson Orin-Nano.

5 Implementation Details

We implemented our approach[1] using the *Flower* framework (Beutel et al., 2020) in *PyTorch* (PyTorch maintainers and contributors, 2016). Pretraining and simulated federated experiments utilized a single NVIDIA L40s GPU, running both the server and active clients in parallel. Real-world experiments employed a server containing a single NVIDIA A100 GPU, and a cluster of 3 Raspberry Pi 4, 3 Raspberry Pi 5, 5 Jetson Nano and 1 Jetson Orin-Nano, all having a minimum of 4GB of RAM (the setup is shown in Figure 3). The supplementary material contains a short video showing the setup and training on real devices.

Lightweight Encoders Pretraining
We pretrain the lightweight encoders on the server for 60 epochs using a batch size of 64. We adopt a cosine-annealing learning rate scheduler (decaying to $lr = 0$) with a peak learning rate of $lr_{\max} = 0.005$. Optimization is performed using Adam ($\beta_1 = 0.9$, $\beta_2 = 0.999$) with no weight decay. We follow the TorchVision benchmark (TorchVision maintainers and contributors, 2016) for MobileNet data augmentation. As a reference, pretraining requires 40 minutes on an NVIDIA L40s for StanfordCars, while timing on other datasets scales roughly with size (see the Appendix).

Simulated Federated Experiments
Optimizer, scheduler, batch size, and peak learning rate are kept the same as in pretraining. The optimization runs for $n_r = 500$ rounds, with $n_e = 10$ local epochs per client (without augmentation). We consider a federated setup with $n_k = 100$ clients, where $n_a = 10$ clients participate at each round. The server-side distillation learning rate is set to 10^{-5} and $\lambda^L_{KD} = 0.01$. Federated finetuning on CompCars requires $\sim$15 hours. The LoRA adapters are applied to each attention layer of the server model, targeting Q and V. The rank is set to 16 for a total of $1.57M$ parameters ($\sim 194\times$ less than the full server model and $1.6\times$ less than the client model). See Table 7 for additional configurations.

On-Device Federated Experiments
The training configuration is analogous to that used in the simulated experiments, the key difference being that no GPU acceleration is employed on the clients. We set the rounds to $n_r = 300$, with $n_e = 10$ local epochs per client (without augmentation). We consider a federated setup with $n_k = 12$ clients, where $n_a = (2, 4)$ clients participate per-round. On-device federated finetuning on CompCars requires $\sim$4 days. The peak learning rate was set to $lr_{\max} = 6 \times 10^{-4}$, and the batch size to 48. The number of active clients was chosen to preserve approximately the same ratio as the simulated experiments (1/10), and the learning rate was chosen to achieve, on average, the same gradient updates.

Datasets
To ensure fairness in the experimental evaluation, we evaluate EFFEKT on the same five domain pairs used by Caligiuri et al. (2025). All datasets exhibit long-tailed class distribution, having an average of 272 target classes belonging to similar objects, and are distributed across clients according to a Dirichlet distribution with concentration parameter $\alpha = 1$ (Hsu et al., 2019). In each scenario, a public source dataset is used for pretraining, while a private target dataset is distributed across clients in a federated setting. More specifically, the dataset couples are organized as follows: CUB200 (Wah et al., 2011) $\rightarrow$ NABirds (Van Horn et al.,

[1]Code will be released upon acceptance.

2015), ImageNetPets (Caligiuri et al., 2025) → OxfordPets (Parkhi et al., 2012), StanfordCars (Krause et al., 2013) → CompCars (Yang et al., 2015), FGVCAircraft (Maji et al., 2013) → MilitaryAircraft (Nakamura, 2024), and Food101 (Bossard et al., 2014) → UECFOOD256 (Kawano & Yanai, 2014).

6 Experimental Results

To evaluate the performance of EFFEKT, we compare it against six competing federated strategies. The first competitor is FedAvg (McMahan et al., 2017), a common federated learning baseline. The second, FedAvg + EMA, is an extension that tackles client divergence by applying Exponential Moving Average (EMA) to the model weights. The third (FedProx (Li et al., 2020)) and fourth (MOON (Li et al., 2021)) competitors are benchmark distributed optimization techniques that actively tackle client divergence by analyzing differences in the received weights. We consider the easiest (higher-accuracy) version of FedProx, where no *straggler* clients are present. As the fifth competitor, we selected the recent FedHEAL (Chen et al., 2024) approach, which extends the standard FedAVG aggregation, introducing a similarity-based re-weighting factor. The last competitor is FedPromo (Caligiuri et al., 2025), which introduced the ICP regularization we deploy at the client-side. Finally, we provide the performance of a centralized configuration as a reference. Table 1 compares the Top-1 and Top-5 classification accuracy of the server model $H_k^{d_i} \circ O^{d_i}$ with those of the competitors. Finally, we report the performance of a centralized experiment implementing the EFFEKT bi-directional distillation. Here, a single client trains its head on the full private dataset $\mathcal{D}^{d_i}$ for $n_r = 500$ rounds, while the server updates the domain-specific LoRAs.

Table 1: Federated Top-1 and Top-5 accuracy (↑) on the server after federated aggregation. **Best** in bold, second-best underlined.

Dataset $\mathcal{D}$	Method	DINOv2 Top-1	DINOv2 Top-5
CompCars	FedAvg	14.0	27.4
	FedAvg + EMA	21.5	42.6
	FedProx	13.7	31.2
	MOON	13.8	27.4
	FedHEAL	13.9	27.0
	FedPromo	35.8	69.4
	EFFEKT (ours)	**43.0**	**75.0**
	Centralized	43.2	73.6
UECFOOD256	FedAvg	29.6	50.4
	FedAvg + EMA	49.1	83.1
	FedProx	8.2	42.4
	MOON	29.7	50.5
	FedHEAL	29.5	50.2
	FedPromo	58.7	88.7
	EFFEKT (ours)	**62.1**	**88.8**
	Centralized	62.8	89.3
NABirds	FedAvg	6.1	12.9
	FedAvg + EMA	20.7	43.6
	FedProx	16.0	39.9
	MOON	6.1	12.9
	FedHEAL	6.2	13.0
	FedPromo	30.3	65.0
	EFFEKT (ours)	**39.1**	**72.8**
	Centralized	37.9	69.2
Military Aircraft	FedAvg	9.2	27.7
	FedAvg + EMA	12.4	**39.0**
	FedProx	10.4	33.8
	MOON	9.6	27.8
	FedHEAL	9.5	27.9
	FedPromo	11.9	32.1
	EFFEKT (ours)	**12.5**	33.7
	Centralized	13.6	34.2
OxfordPets	FedAvg	47.9	79.2
	FedAvg + EMA	67.6	96.4
	FedProx	68.0	95.0
	MOON	47.4	78.8
	FedHEAL	50.1	79.5
	FedPromo	**72.7**	**98.0**
	EFFEKT (ours)	72.3	96.6
	Centralized	80.0	99.2

6.1 Single Domain Results

Our approach surpasses competitors in most of the considered domains. In particular, in the StanfordCars→CompCars scenario, EFFEKT's improvement over the second-best approach (FedPromo) reaches 7.2% and 5.6% Top-1 and Top-5 accuracy, respectively. A similar result is achieved in the Top-1 accuracy of the Food101→UECFOOD256 scenario, where the improvement is 3.4%. Top-5, instead, gains a more limited 0.1%. The highest gains are observed in the CUB200→NABirds scenario, where EFFEKT surpasses competitors by more than 8.8% in Top-1 accuracy and 7.8% in Top-5. The FVGAircraft→MilitaryAircraft scenario is the most complex in our experiments, where results in absolute terms are limited for all competitors. In particular, the highest Top-1 accuracy (12.5%) is achieved by EFFEKT, while the highest Top-5 (39.0%) is achieved by FedAvg+EMA. The last

Table 2: Accuracy on multi-domain scenario. *Known Domain*: query sample includes domain index i. *Unknown Domain:* D used to estimate i (using pretraining in-domain dataset prototypes).

Dataset	Known Domain		Unknown Domain			
	Top-1	Top-5	Top-1		Top-5	
CompCars	43.0	75.0	42.7	− 0.3	74.6	− 0.4
UECFOOD256	62.1	88.8	61.4	− 0.7	87.7	− 1.1
NABirds	39.1	72.8	38.5	− 0.6	71.7	− 0.9
Military Aircraft	12.5	33.7	8.0	− 4.5	20.3	−13.4
OxfordPets	72.3	96.6	71.8	− 0.5	96.0	− 0.6
Average	45.8	73.4	44.5	− 1.3	70.1	− 3.3

Table 3: Accuracy of D in identifying the correct domain using prototypes from pretrain and target domains.

Dataset	$\mathcal{D}_p^{d_i}$-Acc	$\mathcal{D}^{d_i}$-Acc
CompCars	99.7	99.6
UECFOOD256	99.5	98.5
NABirds	98.9	89.7
Military Aircraft	60.0	95.6
OxfordPets	99.2	97.1
Average	91.5	95.1

configuration is the simplest, and all methods achieve remarkable accuracy. EFFEKT ranks second behind FedPromo, achieving a Top-1 accuracy of 72.3% and a Top-5 accuracy of 96.6%. Finally, we remark that EFFEKT closely matches the centralized experiment in most domains, confirming its stability.

6.2 Multi-Domain Results

Table 2 shows the accuracy achieved in the multi-domain scenario. The results in Section 6.1 correspond to the case where the sample's domain is known (*i.e.*, when the server-side FM receives a sample, the domain index is included). If this information is not provided, the domain classifier D must be used to estimate it, which may lead to a decrease in accuracy due to domain misclassification. The Table shows that the impact of this issue is limited, with a drop in Top-1 accuracy of less than 1% in all domains except for MilitaryAircraft. For consistency, we also report the Top-5 accuracy, which exhibits a trend similar to that of Top-1. These results are confirmed by Table 3, which reports the domain classification performance. Using the pretraining datasets prototypes, the domain is recognized correctly with an accuracy of 99% for all domains except MilitaryAircraft, where it drops to 60%. This domain proved harder to recognize, as the images are significantly different from those in FGVCAircraft. In many images, the target aircraft appears small and distant, while other objects occupy the foreground. In FGVCAircraft, instead, there is always a plane as the main object. In the Table, we also report the accuracy when computing prototypes on the target datasets; in this case the accuracy drop on MilitaryAircraft is restored.

6.3 On-device Training Results

Table 4 shows the accuracy achieved by our on-device setup. We performed two different tests, setting the number of active clients n_a to 2 and 4. The results confirm the validity of our simulations, given the tight match between experiments. Further validation experiments are in the Appendix: it shows that the real and simulated validation accuracies match is preserved at all steps of the training evolution (see Fig. A.3). We also measured the power consumption and network usage throughout the federated optimization, as detailed in the Appendix. The former never exceeds 7W and scales linearly with the number of active clients, while the latter never exceeds 5MB/s. In Figures A.1 and A.2, we show both measures over a short time interval.

Table 4: Comparison of on-device and simulated experiments.

n_a / n_k	Simulation		Deployment	
	Top-1	Top-5	Top-1	Top-5
2 / 12	39.5	70.0	39.5	70.4
4 / 12	40.0	70.3	39.8	70.5
Average	39.8	70.2	39.7	70.5

7 Ablation Studies

Component Analysis As a first ablation study, we evaluate the impact of the various components of our architecture on performance using the CompCars dataset. The results are shown in Table 5.

When disabling both C2S and JA Distillations (*i.e.*, using FedAVG for server-side aggregation), the Top-1 accuracy drops by more than 7% (the Top-5 exhibits similar behavior, with a drop of almost 6% as well). The impact of both components on Top-1 accuracy is similar: when using only C2S or only JA, the drop in performance is around 2.5% (from 43% to 40.6% and 40.5%, respectively). This finding demonstrates that both strategies are necessary and complementary; by combining them, we achieve a noticeable improvement. The Top-5 exhibits similar behavior, even though, according to this metric, the C2S strategy has a slightly greater impact. This could stem from the missing cross-entropy objective, which tightens the entropy of the predictions, leading to slightly higher Top-1 accuracy at the cost of Top-5. This is confirmed by the ablation on the loss terms used in JA, where disabling the cross-entropy leads to a higher Top-5 accuracy compared to the other partial configurations. Regardless, the experiments confirm the optimal configuration of our approach, as using all components together yields the highest accuracy. To further validate the effectiveness of our two distillation objectives, in Section A.3 of the *Appendix* we report the accuracy when C2S and JA are not performed at every communication round. Overall, we observe an approximately linear increase in performance with respect to the application frequency $f \in [0, 1]$ (one every R rounds). More specifically, the average performance on the server can be estimated as $\text{acc}_1 \simeq 7.93f + 35.3$ and $\text{acc}_5 \simeq 6.05f + 69.1$ with an R^2 value of 0.97 for both. As an additional study, we report the comparative analysis when swapping the client encoder architecture to EfficientNet-B0 (Tan & Le, 2020). The results are reported in the Appendix (Figures A.7 for Top-1 Accuracy and A.8 for Top-5 Accuracy), confirming a consistent improvement of EFFEKT with respect to all other competitors (approximately 5% improvement over the second best, FedPromo, and 30% over FedAvg).

Table 5: Ablation on the impact of the different modules and losses.

C2S	JA $\mathcal{L}_{CE}$	JA $\mathcal{L}^L_{KD}$	JA $\mathcal{L}^F_{KD}$	Top-1	Top-5
✗	✗	✗	✗	35.8	69.4
✗	✓	✓	✓	40.6	72.6
✓	✗	✗	✗	40.5	73.7
✓	✗	✓	✓	41.9	74.6
✓	✓	✗	✓	39.2	72.7
✓	✓	✓	✗	41.2	72.7
✓	✓	✓	✓	43.0	75.0

Hyperparameters
We also evaluate the impact of setting different values for the algorithm hyperparameters. The balance between feature and logits distillation in Eq. 2 is controlled by the λ^L_{KD} parameter. As shown in Table 6, the proposed value of 0.01 leads to the best performance, while smaller or larger values result in a decrease in performance for both Top-1 and Top-5 accuracy. Another relevant design parameter is the rank for the LoRA adapters (Table 7). Performance increases when using larger rank adapters up to a certain point, but it decreases again after the optimal value is achieved for rank 16. Additionally, we report the performance achieved by training the full FM without employing adapters (*full FM*). The performance drops below that of rank 32, confirming that the decreasing trend continues to full rank. We hypothesize that this is due to the limited number of samples and the misalignment between proxy and server features, which leads to overfitting using larger adapters.

Table 6: Ablation on logit distillation loss weight.

λ^L_{KD}	Top-1	Top-5
0.001	40.6	73.6
0.005	41.5	73.8
<u>0.01</u>	43.0	75.0
0.02	42.2	74.3
0.1	41.7	74.5

Table 7: Ablation on the rank of LoRA adapters.

rank	Top-1	Top-5
8	42.0	74.2
<u>16</u>	43.0	75.0
32	41.5	73.9
full FM	41.3	71.1

Federated Configuration
Finally, we evaluate performance in different federated settings. We compared EFFEKT with the baseline FedAvg approach and the best competitor, *i.e.*, Caligiuri et al. (2025), on the CompCars dataset. We computed results by varying the number of rounds, the Dirichlet concentration parameter, and the total and active client numbers. The detailed results are in the Appendix. Note how our approach consistently outperforms the competitors in all the considered settings. Finally, the

standard deviation of the Top-1 accuracy of EFFEKT, computed by running five experiments with different random seeds, is very limited (0.5%). This is on par with those of the competitors, in spite of the higher mean values. Figures A.5 and A.6 in the Appendix show how this result is consistent across rounds.

8 Conclusions

In this paper, we introduced EFFEKT, a novel multi-domain federated learning framework enabling efficient collaboration between lightweight client-side proxy models and a server-side Foundation Model. EFFEKT's key innovation lies in its bi-directional cross-distillation strategies and the efficient server-side training of domain-specific LoRA adapters. Experiments conducted across multiple real-world datasets demonstrated consistent improvements in Top-1 and Top-5 accuracy compared to competitors. Additionally, we validated EFFEKT's real-world applicability through deployments on compute-constrained devices, confirming its efficiency in computation, energy consumption, and network usage. This demonstrated EFFEKT's suitability for privacy-sensitive applications on personal devices, opening up new possibilities for AI in diverse domains.

Impact Statement

This work aims to advance the field of machine learning and does not introduce immediate or specific societal consequences that require separate emphasis; however, it entails ethical considerations typical of decentralized learning settings. In particular, although raw data remain on client devices, shared model components may still be vulnerable to adversarial attacks that leak sensitive information, underscoring the importance of privacy-preserving techniques. Additionally, decentralized training can encode local biases, which may lead to unfair outcomes if not properly addressed. While the evaluations in this paper are designed to avoid these concerns, caution is still warranted when deploying the proposed methods on sensitive private data, such as facial images, in real-world scenarios.

References

Sara Babakniya, Ahmed Elkordy, Yahya Ezzeldin, Qingfeng Liu, Kee-Bong Song, MOSTAFA EL-Khamy, and Salman Avestimehr. SLoRA: Federated parameter efficient fine-tuning of language models. In *International Workshop on Federated Learning in the Age of Foundation Models in Conjunction with NeurIPS 2023*, 2023. URL `https://openreview.net/forum?id=06quMTmtRV`.

Francesco Barbato, Umberto Michieli, Jijoong Moon, Pietro Zanuttigh, and Mete Ozay. Cross-architecture auxiliary feature space translation for efficient few-shot personalized object detection. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8587–8594. IEEE, 2024.

Daniel J Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Javier Fernandez-Marques, Yan Gao, Lorenzo Sani, Hei Li Kwing, Titouan Parcollet, Pedro PB de Gusmão, and Nicholas D Lane. Flower: A friendly federated learning research framework. *arXiv preprint arXiv:2007.14390*, 2020.

Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *Proceedings of the European Conference on Computer Vision (ECCV), Part VI*, pp. 446–461. Springer, 2014.

Matteo Caligiuri, Francesco Barbato, Donald Shenaj, Umberto Michieli, and Pietro Zanuttigh. FedPromo: Federated lightweight proxy models at the edge bring new domains to foundation models. *arXiv preprint arXiv:2508.03356*, 2025.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9650–9660, 2021.

Hong-You Chen, Cheng-Hao Tu, Ziwei Li, Han Wei Shen, and Wei-Lun Chao. On the importance and applicability of pre-training for federated learning. In *Proceedings of the 11th International Conference on Learning Representations (ICLR)*, 2023. URL `https://openreview.net/forum?id=fWWFv--P0xP`.

Yuhang Chen, Wenke Huang, and Mang Ye. Fair federated learning under domain skew with local consistency and domain diversity. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12077–12086, 2024.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

Marko Harasic, Dennis Lehmann, and Adrian Paschke. Evaluating federated dino's performance on the segmentation task across diverse domains. In *2024 IEEE International Conference on Big Data (BigData)*, pp. 7784–7789, 2024. doi: 10.1109/BigData62323.2024.10825380.

Geoffrey Hinton. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Chang Hongyan, Shejwalkar Virat, Shokri Reza, and Houmansadr Amir. Cronus: Robust and heterogeneous collaborative learning with black-box knowledge transfer. *arXiv preprint arXiv:1912.11279*, 2019.

Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1314–1324, 2019.

Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Eunjeong Jeong, Seungeun Oh, Hyesung Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim. Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data. *arXiv preprint arXiv:1811.11479*, 2018.

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pp. 5132–5143. PMLR, 2020.

Y. Kawano and K. Yanai. Automatic expansion of a food image dataset leveraging existing categories with domain adaptation. In *Proceedings of the ECCV Workshop on Transferring and Adapting Source Knowledge in Computer Vision (TASK-CV)*, 2014.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollar, and Ross Girshick. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4015–4026, October 2023.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 554–561, 2013.

Daliang Li and Junpu Wang. Fedmd: Heterogenous federated learning via model distillation. *arXiv preprint arXiv:1910.03581*, 2019.

Qinbin Li, Bingsheng He, and Dawn Song. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10713–10722, 2021.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems*, 2:429–450, 2020.

Yuxi Liu, Guibo Luo, and Yuesheng Zhu. Fedfms: Exploring federated foundation models for medical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 283–293. Springer, 2024.

Wang Lu, Xixu Hu, Jindong Wang, and Xing Xie. Fedclip: Fast generalization and personalization for clip in federated learning. *IEEE Data Engineering Bulletin*, 2023.

S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pp. 1273–1282. PMLR, 2017.

T. Nakamura. Military aircraft detection dataset. Available: `https://www.kaggle.com/datasets/a2015003713/militaryaircraftdetectiondataset`, 2024.

John Nguyen, Jianyu Wang, Kshitiz Malik, Maziar Sanjabi, and Michael Rabbat. Where to begin? on the impact of pre-training and initialization in federated learning. In *Proceedings of the 11th International Conference on Learning Representations (ICLR)*, 2023. URL `https://openreview.net/forum?id=Mpa3tRJFBb`.

Maxime Oquab, Timothée Darcet, Theo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Russell Howes, Po-Yao Huang, Hu Xu, Vasu Sharma, Shang-Wen Li, Wojciech Galuba, Mike Rabbat, Mido Assran, Nicolas Ballas, Gabriel Synnaeve, Ishan Misra, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Oleksiy Ostapenko, Timothee Lesort, Pau Rodriguez, Md Rifat Arefin, Arthur Douillard, Irina Rish, and Laurent Charlin. Continual learning with foundation models: An empirical study of latent replay. In *Conference on lifelong learning agents*, pp. 60–91. PMLR, 2022.

Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3498–3505. IEEE, 2012.

PyTorch maintainers and contributors. Pytorch framework [software]. Available: `https://github.com/pytorch/pytorch`, 2016.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PMLR, 2021.

Chao Ren, Han Yu, Hongyi Peng, Xiaoli Tang, Bo Zhao, Liping Yi, Alysa Ziying Tan, Yulan Gao, Anran Li, Xiaoxiao Li, et al. Advances and open challenges in federated foundation models. *IEEE Communications Surveys & Tutorials*, 2025.

Felix Sattler, Tim Korjakow, Roman Rischke, and Wojciech Samek. Fedaux: Leveraging unlabeled auxiliary data in federated learning. *IEEE Transactions on Neural Networks and Learning Systems*, 34(9):5531–5543, 2021.

Donald Shenaj, Eros Fanì, Marco Toldo, Debora Caldarola, Antonio Tavera, Umberto Michieli, Marco Ciccone, Pietro Zanuttigh, and Barbara Caputo. Learning across domains and devices: Style-driven source-free domain adaptation in clustered federated learning. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 444–454, 2023a.

Donald Shenaj, Giulia Rizzoli, and Pietro Zanuttigh. Federated learning in computer vision. *IEEE Access*, 2023b.

Jost Tobias Springenberg. Unsupervised and semi-supervised learning with categorical generative adversarial networks. *arXiv preprint arXiv:1511.06390*, 2015.

Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2020. URL `https://arxiv.org/abs/1905.11946`.

TorchVision maintainers and contributors. Torchvision: Pytorch's computer vision library [software]. Available: `https://github.com/pytorch/vision`, 2016.

Grant Van Horn, Steve Branson, Ryan Farrell, Scott Haber, Jessie Barry, Panos Ipeirotis, Pietro Perona, and Serge Belongie. Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. *Technical Report CNS-TR-2011-001, Computation & Neural Systems, Caltech*, 2011.

Zhiyuan Wu, Sheng Sun, Yuwei Wang, Min Liu, Quyang Pan, Junbo Zhang, Zeju Li, and Qingxiang Liu. Exploring the distributed knowledge congruence in proxy-data-free federated distillation. *ACM Transactions on Intelligent Systems and Technology*, 15(2):1–34, 2024.

Linjie Yang, Ping Luo, Chen Change Loy, and Xiaoou Tang. A large-scale car dataset for fine-grained categorization and verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3973–3981, 2015.

Dezhong Yao, Wanning Pan, Yutong Dai, Yao Wan, Xiaofeng Ding, Chen Yu, Hai Jin, Zheng Xu, and Lichao Sun. Fedgkd: Toward heterogeneous federated learning via global knowledge distillation. *IEEE Transactions on Computers*, 73(1):3–17, 2023.

Liping Yi, Han Yu, Gang Wang, Xiaoguang Liu, and Xiaoxiao Li. pfedlora: Model-heterogeneous personalized federated learning with lora tuning. *arXiv preprint arXiv:2310.13283*, 2023.

Liangqi Yuan, Ziran Wang, Lichao Sun, Philip S Yu, and Christopher G Brinton. Decentralized federated learning: A survey and perspective. *IEEE Internet of Things Journal*, 11(21):34617–34638, 2024.

Lan Zhang, Dapeng Wu, and Xiaoyong Yuan. Fedzkt: Zero-shot knowledge transfer towards resource-constrained federated learning with heterogeneous on-device models. In *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*, pp. 928–938. IEEE, 2022.

Zhuangdi Zhu, Junyuan Hong, and Jiayu Zhou. Data-free knowledge distillation for heterogeneous federated learning. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 12878–12889. PMLR, 18–24 Jul 2021. URL `https://proceedings.mlr.press/v139/zhu21b.html`.

A Appendix

In the following sections, we report the additional and supporting experiments that could not fit in the main document due to space limitations. We begin by providing experiments varying all aspects of our Federated Learning training configuration in Section A.1; we then move to Section A.2 that reports our power and network consumption measurement setup, as well as our findings; Section A.3 reports some additional quantitative experiments and some training evolution curves, highlighting the stability of out approach; finally, in Section A.4 we report the pseudocode implementation for the remaining modules of EFFEKT.

A.1 Federated Configuration

This section contains additional experiments varying the Federated Learning configuration, comparing the accuracy of EFFEKT, FedPromo, and FedAvg. In particular, Table A.1 focuses on the number of rounds, Table A.2 on the number of total clients, Table A.3 on the number of active clients, and Table A.4 on the Dirichlet concentration parameter α. The results show that the configuration reported in the main document is optimal, leading to the best performance even when changing scenarios, *e.g.*, using fewer clients, or more active clients. We hypothesize that this depends on the bi-directional distillation hyperparameters, which have been tuned with the target configuration.

Regarding the number of rounds, the EFFEKT results behave as expected: the shorter the training, the lower the accuracy. This contrasts with FedPromo, where accuracy remains stable across 250–500 rounds, and with FedAvg, where shorter training actually leads to better performance. The likely reason is that the LoRA adapters in EFFEKT require a relatively larger number of steps to be properly optimized, while the performance drop in FedAvg is caused by client divergence, which FedPromo effectively mitigates.

Interestingly, EFFEKT attains higher accuracy even for smaller values of the Dirichlet concentration parameter α. This advantage stems from our C2S distillation, which is able to extract useful information from all client updates (including slightly divergent ones). As a result, smaller α values lead to more informative update directions, whereas larger values tend to pull the updates into a similar direction, thereby diminishing their information content. Note how EFFEKT continues to be the best-performing method across all settings; in this experiment even its lowest accuracy surpasses the highest accuracy achieved by FedPromo.

A.2 Power and Network Consumption

As noted in the main document, we deploy EFFEKT on a cluster of 12 compute-constrained real devices, connected to the server and to each other via a LAN network. More specifically, to measure their power consumption, we clustered the clients by manufacturer (Raspberry Pi Foundation vs. NVIDIA), plugging each group into a different Sonoff S31 (Shenzhen Sonoff Technologies Co., Ltd., 2015) smart power plug (running

Table A.1: Number of rounds.

n_r	FedAvg		FedPromo		EFFEKT	
	top-1	top-5	top-1	top-5	top-1	top-5
100	22.1	41.9	30.6	63.9	30.8	63.3
250	18.4	33.2	35.3	69.1	37.9	70.9
<u>500</u>	14.0	27.4	35.8	69.4	43.0	75.0

Table A.2: Number of total clients.

n_k	FedAvg		FedPromo		EFFEKT	
	top-1	top-5	top-1	top-5	top-1	top-5
50	10.6	22.5	36.2	68.4	40.8	72.4
<u>100</u>	14.0	27.4	35.8	69.4	43.0	75.0
150	12.8	24.0	29.6	59.3	39.0	70.5

Table A.3: Number of active clients.

n_a	FedAvg		FedPromo		EFFEKT	
	top-1	top-5	top-1	top-5	top-1	top-5
5	11.0	21.9	36.9	70.1	41.6	73.3
<u>10</u>	14.0	27.4	35.8	69.4	43.0	75.0
20	10.8	21.5	37.2	70.7	42.1	74.4

Table A.4: Dirichlet concentration parameter.

α	FedAvg		FedPromo		EFFEKT	
	top-1	top-5	top-1	top-5	top-1	top-5
<u>1</u>	14.0	27.4	35.8	69.4	43.0	75.0
10	12.0	24.0	37.1	70.4	42.4	74.4
100	12.5	24.7	37.3	70.1	42.1	74.0

TASMOTA firmware (Tasmota maintainers and contributors, 2017)). In the first group, we included 3 Raspberry Pi 4 and 3 Raspberry Pi 5; in the second group, we had 5 Jetson Nano and 1 Jetson Orin-Nano. Logging was performed using an open source MQTT broker (Mosquitto (Eclipse Foundation, 2009)) and a Python script that polls the server every second. Regarding the network statistics, to avoid affecting the measurements, we deployed a Grafana dashboard (Grafana Labs, 2013) on a separate machine (distinct from the server and clients) that receives information from a Prometheus service (Prometheus maintainers and contributors, 2013) running on each client via a separate network interface.

We measure an average power consumption across the federated optimization of 2.1W and 3.6W by the Raspberry Pi devices, under $n_a = 2$ and $n_a = 4$, respectively. The NVIDIA boards absorb approximately 70% more power, landing at 3.3W and 6.3W under the same configurations. In Figure A.1, we plot the consumption on both configurations over a small time slice. The spikiness of the plot is a consequence of the active client sampling of the server. A full training with 2 active clients absorbs $\sim$ 3.5kWh, split in 1.5kWh for Jetson and 2kWh for Raspberry (sampled more often, as they are faster).

A similar plot, showing the network usage over a time window, is reported in Figure A.2. Most of the clients' throughput is in downloads since they need to update the full local model (encoder and head) at each round, but they only send the head to the server. This is confirmed by the global statistics, which never exceed 4MB/s in RX, and 60kB/s in TX. Moreover, on average each client transmits 3.7MB (which closely matches the size of $H_k^{d_i}$) and receives 47.9MB.

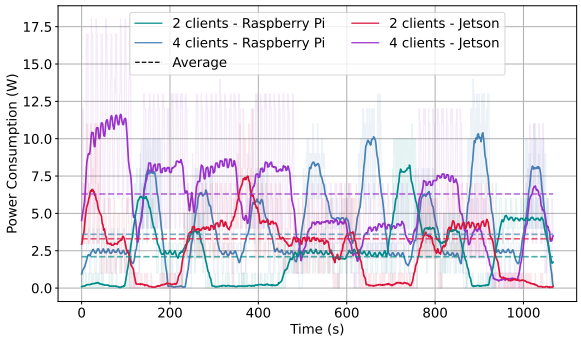

Figure A.1: Training power consumption (short time window).

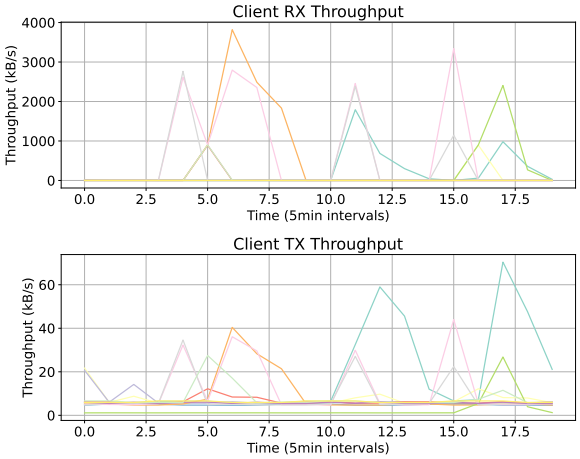

Figure A.2: Training network usage (short time window).

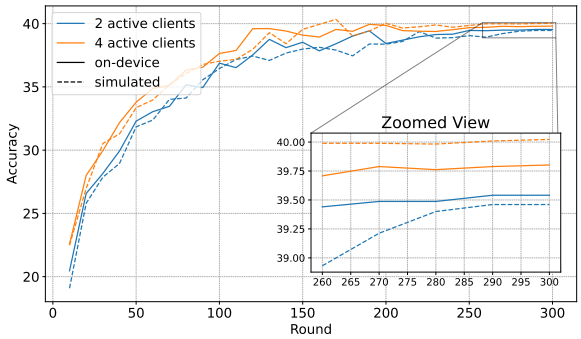

Figure A.3: Simulated and real devices, Top-1 accuracy.

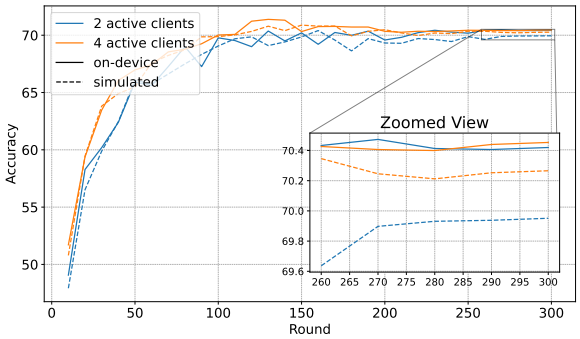

Figure A.4: Simulated and real devices, Top-5 accuracy.

A.3 Additional Experiments

Figures A.3 and A.4 report the Top-1 and Top-5 accuracy achieved by EFFEKT on the CompCars dataset under configurations with 2 and 4 active clients, comparing simulated experiments against real-device deployments. The results obtained on real hardware closely match those observed in simulation, providing strong evidence for the reliability of our simulated setup. As expected, increasing the number of active clients from 2 to 4 leads to higher accuracy; however, the performance gap remains relatively small. This limited difference highlights the robustness and stability of EFFEKT, which is enabled by its bidirectional distillation mechanism and allows the framework to perform consistently even with fewer participating clients.

Furthermore, Figures A.5 and A.6 illustrate the training evolution of Top-1 and Top-5 accuracy across communication rounds, including a shaded region representing the confidence interval computed over 5 independent runs with different random seeds. EFFEKT consistently outperforms all competing methods throughout training, exceeding the closest baseline, FedPromo, by more than 5% in both metrics. Notably, the confidence intervals (for both Top-1 and Top-5) of EFFEKT never overlap with those of the other approaches, providing clear evidence that the observed improvements are statistically significant and robust.

As additional validation, in Figures A.7 and A.8 we analyze training dynamics when changing the client-side architecture, from MobileNetV3-Small to EfficientNet-B0, and compare EFFEKT with FedPromo and FedAvg.

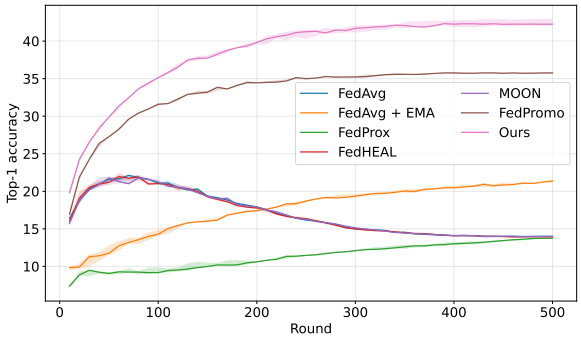

Figure A.5: Validation server top-1 accuracy evolution.

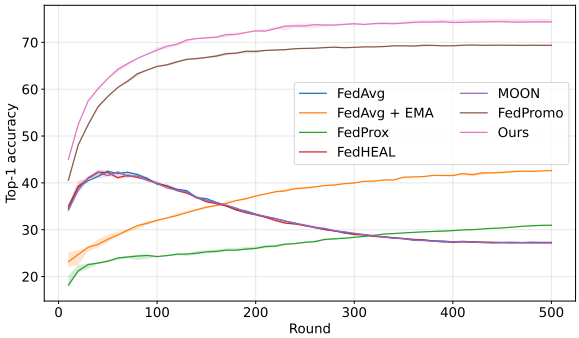

Figure A.6: Validation server top-5 accuracy evolution.

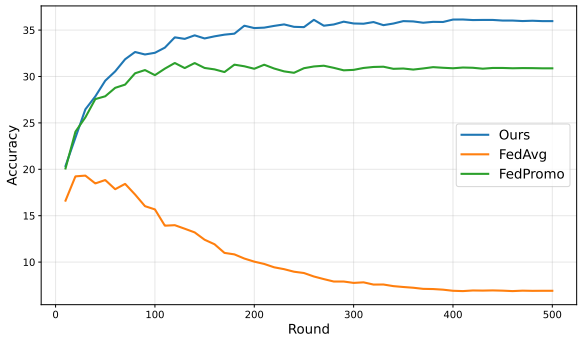

Figure A.7: Validation server top-1 accuracy evolution using EfficientNet-B0 as client encoder.

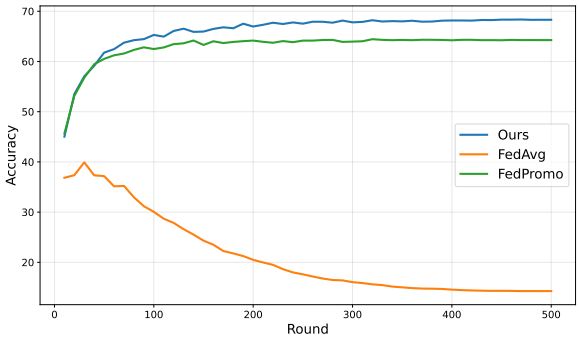

Figure A.8: Validation server top-5 accuracy evolution using EfficientNet-B0 as client encoder.

In this setting, FedAvg collapses in both Top-1 and Top-5 accuracy, whereas FedPromo and EFFEKT remain stable across training rounds. Despite EfficientNet-B0 having a larger number of parameters and relying on a different architectural design compared to the standard classification head, overall performance is lower than in previous experiments.

Nevertheless, EFFEKT maintains a clear advantage, achieving approximately a 5% improvement over FedPromo in both accuracy metrics, further demonstrating its robustness to architectural changes on the client side.

In Table A.5, we report results analogous to those presented in Table 2 of the main paper, with the key difference that the prototypes of the domain classifier are computed using the private target-domain data rather than the task-specific pretraining datasets.

Overall, the results closely match those obtained in the main table, thereby further validating the effectiveness and consistency of the proposed approach. The only notable deviation is observed on the MilitaryAircraft dataset, where performance improves substantially. This gain can be attributed to the removal of the domain shift between MilitaryAircraft and its previously coupled domain, FGVCAircraft, when prototypes are derived directly from the target data. It is important to emphasize, however, that these results should be interpreted strictly as upper bounds, since in a realistic federated deployment, the server does not have access to the clients' private target-domain data.

Furthermore, Table A.6 presents companion results to Table 1 in the main document, obtained by replacing the multiple task-specific pretraining datasets with a single generic dataset, ImageNet. In this setting, performances degrade significantly across most domains. This drop highlights that the client-side encoder alone is not sufficiently expressive to accurately approximate the server-side foundation model, leading to increased representation misalignment during federated training. Despite this limitation, EFFEKT achieves performances that are largely comparable to both FedPromo and centralized training in most configurations. Absolute accuracy remains

Table A.5: Accuracy on multi-domain scenario. *Domain Knowledge*: query sample includes domain index i. *Unknown Domain:* D used to estimate i.

Dataset	Domain Knowledge		Unknown Domain	
	Top-1	Top-5	Top-1	Top-5
CompCars	43.0	75.0	42.7 − 0.3	74.6 − 0.4
UECFOOD256	62.1	88.8	60.6 − 1.5	86.9 − 1.9
NABirds	39.1	72.8	33.7 − 5.4	62.7 − 10.1
Military Aircraft	12.5	33.7	11.8 − 0.7	31.7 − 2.0
OxfordPets	72.3	96.6	70.5 − 1.8	94.0 − 2.6
Average	45.8	73.4	43.9 − 1.9	70.0 − 3.4

Table A.6: Out-of-domain experiments. Federated Top-1 and Top-5 accuracy (↑) starting from ImageNet-1k (1000 classes) pretraining. **Best** in bold, second-best underlined.

Dataset $\mathcal{D}$	Method	DINOv2	
		Top-1	Top-5
CompCars	FedAvg	1.2	3.9
	FedAvg + EMA	3.0	9.7
	FedProx	2.4	8.5
	MOON	1.3	3.9
	FedHEAL	1.2	4.0
	FedPromo	**3.3**	**10.8**
	EFFEKT (ours)	2.9	10.1
	Centralized	3.3	9.8
UECFOOD256	FedAvg	2.1	6.9
	FedAvg + EMA	23.5	48.4
	FedProx	4.0	13.2
	MOON	2.0	7.1
	FedHEAL	2.1	7.0
	FedPromo	24.0	48.5
	EFFEKT (ours)	**24.9**	**49.4**
	Centralized	26.1	52.5
NABirds	FedAvg	1.5	5.1
	FedAvg + EMA	11.3	29.1
	FedProx	10.5	26.6
	MOON	1.6	5.0
	FedHEAL	1.4	4.9
	FedPromo	**18.5**	**48.6**
	EFFEKT (ours)	16.5	44.5
	Centralized	16.7	43.0
Military Aircraft	FedAvg	5.1	16.2
	FedAvg + EMA	8.0	**25.8**
	FedProx	**8.3**	24.8
	MOON	5.3	16.2
	FedHEAL	5.1	16.5
	FedPromo	7.1	24.1
	EFFEKT (ours)	8.0	24.6
	Centralized	9.2	28.8
OxfordPets	FedAvg	84.0	99.5
	FedAvg + EMA	85.2	99.8
	FedProx	85.6	**99.9**
	MOON	84.7	99.5
	FedHEAL	84.5	99.5
	FedPromo	**89.5**	99.8
	EFFEKT (ours)	89.0	**99.9**
	Centralized	88.5	99.9

low for nearly all domains, with the notable exception of OxfordPets, which is effectively task-specific with respect to ImageNet, as its samples are drawn from the same validation set.

Finally, in Table A.7, we report the final server accuracy when our two distillation techniques (C2S and JA) are not performed at every communication round. The results suggest a roughly linear relationship between the frequency of the distillation and the server performance. More specifically, we report the performance when C2S and JA are enabled with frequency $f = \{1, \frac{1}{2}, \frac{1}{5}, \frac{1}{10}, 0\}$ rounds. Linear fitting attains the following relationship between the two metrics: $\text{acc}_1 \simeq 7.93f + 35.3$ and $\text{acc}_5 \simeq 6.05f + 69.1$ with an R^2 value of 0.97 for both.

Frequency	Top-1	Top-5
0 (FedPromo)	35.8	69.4
1/10	35.4	69.0
1/5	36.6	70.5
1/2	40.0	72.5
1 (EFFEKT)	43.0	75.0

Table A.7: Accuracy vs. distillation frequency.

A.4 EFFEKT Pseudocode

In this section, we report the pseudocode implementation of EFFEKT. We begin by detailing the EFFEKT domain identification pipeline in Algorithm A.1. We then report a detailed description of the EFFEKT ClientRound (implementing the ICP regularization (Caligiuri et al., 2025)) in Algorithm A.2, and of our two main components (C2S and JA) in Algorithms A.3 and A.4.

Algorithm A.1 EFFEKT Multi-Domain Server Inference.

Require: Server encoder $\hat{O}$, domain classifier D, domain LoRAs $\{L^{d_i}\}_{i \in \mathcal{I}}$, domain heads $\{H_k^{d_i}\}_{i \in \mathcal{I}}$
Input: Input image $\mathbf{X}$
$i \leftarrow D(\mathbf{X})$ // Compute domain index
$O^{d_i} \leftarrow \text{ApplyLoRA}\left(\hat{O}, L^{d_i}\right)$ // Apply correct LoRA
$\mathbf{F}^O \leftarrow O^{d_i}(\mathbf{X})$ // Compute features
$\mathbf{L}^O \leftarrow H^{d_i}(\mathbf{X})$ // Compute logits
$\tilde{y} \leftarrow \text{ArgMax}_{y \in \mathcal{Y}}(\mathbf{L}^O[y])$ // Compute prediction
Output: $\tilde{y}$

Algorithm A.2 EFFEKT ClientRound.

Require: Number of local epochs n_e, domain index i, client index j, batch size b, encoder E^{d_i}
Input: Previous round's head $H_k^{d_i}$, and current lr
Initialize local model $M_{k_j}^{d_i} \leftarrow H_k^{d_i} \circ E^{d_i}$
for $e \leftarrow 1 \dots n_e$ **do**
  $M_{k_j}^{d_i} \leftarrow \text{epochICP}\left(M_{k_j}^{d_i}, b, lr, e\right)$ // See Caligiuri et al. (2025) for details
end for
Extract current head $H_{k_j}^{d_i}$ from $M_{k_j}^{d_i}$
Send: $H_{k_j}^{d_i}$ to server

Algorithm A.3 EFFEKT Clients-to-Server distillation.

Require: Domain index i, server encoder $\hat{O}$, pretraining dataset $\mathcal{D}_p^{d_i}$, batch size b

Input: LoRA L^{d_i}, previous round's encoder E^{d_i}, updated heads from clients $\mathcal{K}_a^{d_i}$: $\mathcal{H}^{d_i} = \{H_{k_j}^{d_i}\}_{j \in \mathcal{K}_a^{d_i}}$

$O^{d_i} \leftarrow \text{ApplyLoRA}\left(\hat{O}, L^{d_i}\right)$
for $q \leftarrow 1 \dots \lfloor |\mathcal{D}_p^{d_i}|/b \rfloor$ **do**
  Extract b samples from dataset $\mathcal{D}_p^{d_i}$ into $\mathcal{B}$
  $l \leftarrow 0$ // Initialize batch loss
  for $(\mathbf{X}, y) \in \mathcal{B}$ **do**
    $\mathbf{F}^O \leftarrow O^{d_i}(\mathbf{X})$ // Extract Features
    $\mathbf{F}^E \leftarrow E^{d_i}(\mathbf{X})$
    for $H^{d_i} \in \mathcal{H}^{d_i}$ **do**
      $\mathbf{L}^O \leftarrow H^{d_i}(\mathbf{F}_O)$ // Compute Logits
      $\mathbf{L}^E \leftarrow H^{d_i}(\mathbf{F}_E)$
      $l \leftarrow l + \mathcal{L}_{KD}^L(\mathbf{L}_O, \mathbf{L}_E)$ // Accumulate loss
    end for
  end for
  $L^{d_i} \leftarrow \text{AdamOptim}(L^{d_i}, l)$ // Update LoRA
end for

Output: L^{d_i}

Algorithm A.4 EFFEKT Joint Alignment distillation.

Require: Domain index i, server encoder $\hat{O}$, pretraining dataset $\mathcal{D}_p^{d_i}$, batch size b, logit KD loss weight λ_{KD}^L

Input: Aggregated client model $M_k^{d_i}$, LoRA L^{d_i}, pretraining head $H_p^{d_i}$

$O^{d_i} \leftarrow \text{ApplyLoRA}\left(\hat{O}, L^{d_i}\right)$
$H_k^{d_i} \circ E^{d_i} = M_k^{d_i}$ // Split model in encoder and head
for $q \leftarrow 1 \dots \lfloor |\mathcal{D}_p^{d_i}|/b \rfloor$ **do**
  Extract b samples from dataset $\mathcal{D}_p^{d_i}$ into $\mathcal{B}$
  $l_{KD}^L \leftarrow 0$ // Initialize batch logits KD loss
  $l_{KD}^F \leftarrow 0$ // Initialize batch feature KD loss
  $l_{CE} \leftarrow 0$ // Initialize batch cross entropy loss
  for $(\mathbf{X}, y) \in \mathcal{B}$ **do**
    $\mathbf{F}^O \leftarrow O^{d_i}(\mathbf{X})$ // Extract features
    $\mathbf{F}^E \leftarrow E^{d_i}(\mathbf{X})$
    $\mathbf{L}^O \leftarrow H_k^{d_i}(\mathbf{F}^O)$ // Client head logits
    $\mathbf{L}^E \leftarrow H_k^{d_i}(\mathbf{F}^E)$
    $\mathbf{L}_p^O \leftarrow H_p^{d_i}(\mathbf{F}^O)$ // Pretraining head logits
    $\mathbf{L}_p^E \leftarrow H_p^{d_i}(\mathbf{F}^E)$
    $l_{KD}^F \leftarrow l_{KD}^F + \mathcal{L}_{KD}^F(\mathbf{F}^O, \mathbf{F}^E) + \mathcal{L}_{KD}^F(\mathbf{F}^E, \mathbf{F}^O)$
    $l_{KD}^L \leftarrow l_{KD}^L + \mathcal{L}_{KD}^L(\mathbf{L}^O, \mathbf{L}^E) + \mathcal{L}_{KD}^L(\mathbf{L}^E, \mathbf{L}^O)$
    $l_{CE} \leftarrow l_{CE} + \mathcal{L}_{CE}(\mathbf{L}_p^O, y) + \mathcal{L}_{CE}(\mathbf{L}_p^E, y)$
  end for
  $l \leftarrow l_{CE} + l_{KD}^F + \lambda_{KD}^L \cdot l_{KD}^L$ // Merge losses
  $M_k^{d_i} \leftarrow H_k^{d_i} \circ E^{d_i}$ // Merge client model
  $\left(L^{d_i}, M_k^{d_i}, H_p^{d_i}\right) \leftarrow \text{AdamOptim}(L^{d_i}, M_k^{d_i}, H_p^{d_i}, l)$ // Update Models
end for

Output: $\left(M_k^{d_i}, L^{d_i}, H_p^{d_i}\right)$

References

Matteo Caligiuri, Francesco Barbato, Donald Shenaj, Umberto Michieli, and Pietro Zanuttigh. FedPromo: Federated lightweight proxy models at the edge bring new domains to foundation models. *arXiv preprint arXiv:2508.03356*, 2025.

Eclipse Foundation. Eclipse mosquitto [software]. Available: `https://mosquitto.org/`, 2009.

Grafana Labs. Graphana dashboard [software]. Available: `https://grafana.com/`, 2013.

Prometheus maintainers and contributors. Prometheus [software]. Available: `https://prometheus.io`, 2013.

Shenzhen Sonoff Technologies Co., Ltd. Sonoff [hardware]. Available: `https://sonoff.tech/`, 2015.

Tasmota maintainers and contributors. Tasmota firmware [software]. Available: `https://tasmota.github.io/docs/`, 2017.