HePCo: Data-Free Heterogeneous Prompt Consolidation for Continual Federated Learning

Shaunak Halbe* **James Seale Smith** **Junjiao Tian** **Zsolt Kira**
Georgia Institute of Technology

Abstract

In this paper, we focus on the important yet understudied problem of Continual Federated Learning (CFL), where a server communicates with a set of clients to incrementally learn new concepts over time without sharing or storing any data. The complexity of this problem is compounded by challenges from both the Continual and Federated Learning perspectives. Specifically, models trained in a CFL setup suffer from catastrophic forgetting which is exacerbated by data heterogeneity across clients. Existing attempts at this problem tend to impose large overheads on clients and communication channels or require access to stored data which renders them unsuitable for real-world use due to privacy. We study this problem in the context of Foundation Models and showcase their effectiveness in mitigating forgetting while minimizing overhead costs and without requiring access to any stored data. We achieve this by leveraging a prompting based approach and proposing a novel and lightweight generation and distillation scheme to aggregate client models at the server. Our approach outperforms both existing methods and our own baselines by more than 7% on challenging image-classification benchmarks while significantly reducing communication and client-level computation costs.

1 Introduction

Federated Learning (FL) is a privacy-preserving learning paradigm that enables learning a global model through communication with a distributed set of clients. These clients have exclusive access to private data, and collaborate with a central server to learn a shared task by communicating parameters such as model weights, gradients, or learning statistics. For example, the popular FedAvg [1] method works by iteratively aggregating client models by averaging model weights.

However, currently most federated learning methods focus on learning statically, that is across a fixed set of categories determined *a-priori*. In non-federated works, on the other hand, there has been a great deal of progress on learning an increasing number of categories incrementally, referred to as *continual* learning (and more specifically *class-incremental learning*) [2, 3]. In addition to the problem of catastrophic forgetting, incremental learning breaks current assumptions in FL, namely that the data is Independent and Identically Distributed (IID), has been shown to cause issues of model divergence [4, 5]. While *heterogeneous federated learning* [5] approaches have been developed, they do not support the *dynamic data distributions* that occur in continual learning and the real-world. Such a setting has immense practical impact and finds direct applications in healthcare, autonomous vehicles, and chat-bots.

Therefore, in this paper we look at the understudied problem of *Continual Federated Learning (CFL)* [6, 7, 8]. While a few CFL methods exist, they often communicate full model weights, real/synthesized image-level data, or gradients. Additionally, some methods store old data in memory buffers or train a generative model to mimic local data; at the very least, all methods share complete

*Correspondence to shalbe9@gatech.edu.

Workshop on Distribution Shifts, 37th Conference on Neural Information Processing Systems (NeurIPS 2023).

models parameters with the server which can lead to privacy leaks with advancements in model inversion and other extraction techniques [9]. *As a result, many of these methods fail to effectively uphold the principles of CFL, such as communication efficiency, computational efficiency and privacy*.

To mitigate forgetting while adhering to the core principles of CFL, we propose HePCo: **H**eterogeneous **P**rompt **Co**nsolidation. Our method is driven by the goals of (i) minimizing communication costs, (ii) improving client privacy, and (iii) client-level computation efficiency. We first propose to leverage *prompting*-based methods, which have shown successful results in the rehearsal-free continual learning setting. This also has the benefit of utilizing frozen Foundation Models, meaning that only prompts and classifiers have to be transmitted, reducing communication costs. The key contribution of our approach is then to answer the question of how to merge prompts from different clients in a scalable manner. Towards this end, we propose a lightweight method for generating pseudo-data and distilling client model information.

2 Related Work

Continual Federated Learning methods currently suffer from various limitations in terms of performance, efficiency and privacy. FedWeiT [6] aims to learn better client models by minimizing interference across client weights. FedWeiT incurs considerable overheads in terms of communication, computation and storage. GLFC [10] uses a prototype based approach with a memory buffer to store old data. This poses a threat to privacy of client data. CFed [7] proposes a distillation based approach that makes use of an unlabelled surrogate dataset to aggregate client models as well as to rehearse old tasks. However the requirement for a curated dataset can severely impact real-world applicability. TARGET [11] combats forgetting through a generative replay of images from past tasks. However, CFed and TARGET introduce overheads at both client and server sides, which may not be ideal for some practical scenarios. In contrast, our approach prioritizes communication efficiency and client privacy by generating in the latent space.

3 Problem Formulation

We focus on the *class-incremental* learning scenario, where a model is tasked with learning new classes over time. Under this setting, a global model is learned through a sequence of N *global* tasks $T = \{T^1, T^2, ..., T^N\}$. As this is done in a federated setup, each task is learned through R independent rounds by randomly sampling a set of *stateless* clients $C = \{c_1, c_2, c_3, ..., c_S\}$ in each round. In a *stateless* setting, new clients are visited in each round. Additionally, to simulate a real-world heterogeneous system, we use three configuration parameters to control the level of heterogeneity with increasing granularity: *split ratio*, *category ratio*, and *imbalance ratio*. *Split ratio* γ is computed as the ratio of the local dataset size to the current task dataset. We denote κ as the *category ratio* which is the ratio of the number of categories each client sees to the total categories in the current task. Finally, we use *imbalance ratio* β to govern an category-level artificial long tailed distribution similar to [12]. If $\beta = 1$, each client c_i is allocated samples uniformly from categories in the current task. In summary, a smaller split ratio γ, a smaller category ratio κ, or a smaller imbalance ratio β increases heterogeneity thereby increasing the complexity of the task.

4 Method

In this section, we describe our novel approach called HePCo (**H**eterogenous **P**rompt **Co**nsolidation) which tackles forgetting and heterogeneity using a data-free distillation strategy applied in the model's latent space. Unlike prior CFL works, we first propose to leverage the current state of art *prompting methods* in continual learning. Such methods optimize learnable parameters that augment the input to a pretrained transformer model (prompt tuning) or its underlying attention mechanism (prefix tuning). These methods have been shown to obtain strong performance without requiring rehearsal. Our key novelty is to propose a lightweight *data-free distillation method*, performed in the latent-space of the model, which greatly mitigates intra-task and inter-task forgetting. Doing so, we prioritize privacy and efficiency, which are crucial for federated learning. Below we detail our method and depict it in Fig. 1.

4.1 Client Side : Decomposed Prompting

L2P (Learning to Prompt) [13] is a continual learning method that maintains a prompt pool $P = \{P_1, P_2, \cdots, P_M\}$ of size M, where $P_i \in \mathbb{R}^{L_p \times D}$ are prompt parameters with L_p as the prompt length (chosen as a hyperparameter) and D the embedding dimension. Each prompt P_i has an associated key $k_i \in \mathbb{R}^{\mathbb{D}}$. An input image x is converted into a visual query $q(x) \in \mathbb{R}^{\mathbb{D}}$ by passing through the frozen vision transformer encoder θ_{pt}. Prompts are selected from the pool by measuring the cosine similarity between associated keys and the visual query to be inserted into the transformer.

While L2P is quite successful in protecting against forgetting, it uses discrete prompts that restrict capacity and introduces an additional hyperparameter (given by N). Instead, we form our final prompt $\boldsymbol{p}$ by taking sum of the individual prompts P_i weighted by the cosine scores. This allows us to effectively learn end-to-end, different from the decoupled optimization in L2P. Our approach only requires the key, prompt, and classifier weights to be transmitted, significantly reducing communication costs compared to sharing complete models. This also safeguards privacy by preventing the server from replicating client models since it has no knowledge of the specific layers where these prompts need to be inserted.

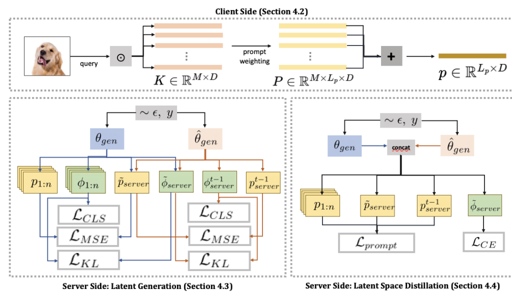

Figure 1: Latent generation and distillation with underlying *decomposed prompting* scheme.

4.2 Server Side : Latent Generation

At the end of each round, We generate pseudo data in the latent space of the *visual query* $q(x) \in \mathbb{R}^{\mathbb{D}}$ which is essentially the output space of the vision encoder. The advantage of generating in this space is that it allows us to fine-tune *both* the classifier and the key-prompt weights without needing a forward-pass through the encoder! We use a lightweight feedforward neural network as our conditional generator with a D dimensional output. From the generator, we obtain a pseudo latent z of dimension D conditioned on the class label as follows: For effective knowledge distillation, pseudo data should conform to the latent space of the client models. We optimize for a classification loss that can be given as: $\mathcal{L}_{cls} = \sum_{c \in C} \mathcal{L}_{CE}(\phi(z; w_c), y)$. Here, ϕ denotes the classifer (last layer). However, optimizing for just the classification loss encourages the generator to produce pseudo latents which are easy to be classified and hence less effective for distillation. To promote the generation of *hard samples*, we maximize two disagreement losses (one for prompts and one for classifier) between server and client models. For classifiers, we compute the Kullback-Leibler (KL) divergence between the predictions of the intermediate server model and each individual client model and for the prompting mechanism, we introduce a Mean-Squared Error (MSE) loss between the final prompts generated by the server and all clients. This is given as : $\mathcal{L}_{KL} = \sum_{c \in C} \sigma(\phi(z; w)) || \sigma(\phi(z; w_c))$ and $\mathcal{L}_{MSE} = \sum_{c \in C} \mathcal{L}_{MSE}(\rho(z; w), \rho(z, w_c))$ where σ denotes the softmax function and ρ denotes the prompting mechanism described in 4.1. We train the generator by optimizing for these these losses jointly as: $\min_{\theta_{gen}} \mathbb{E}_{\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{1})} \left[\mathcal{L}_{cls} - \lambda_{KL} \mathcal{L}_{KL} - \lambda_{MSE} \mathcal{L}_{MSE} \right]$.
A model fine-tuned with only current task pseudo-data suffers from inter-task forgetting as shown in Appendix B. To prevent this, we generate latents corresponding to the previously seen tasks as well.

4.3 Server Side : Latent Space Knowledge Distillation

We use the key, prompt and classifier weights corresponding to the current round client models and the last-task server model to fine-tune the server model. As it operates in a low dimensional latent space, this distillation process is computationally cheap compared to traditional distillation that trains the entire network. We prioritize client-level efficiency while making efficient use of the server's compute resources. To perform knowledge distillation, we first generate a batch of pseudo-data from the generators corresponding to the current round and previous task. We mix the current and previous task batches to form a single composite batch according to a hyperparameter named *replay ratio* which determines the size of the previous task batch relative to the current round batch.

Table 1: **Results (%)** for the class-balanced setups reported over 3 independent trials.

Datasets ($\beta = 1$)	CIFAR-100		ImageNet-R		DomainNet	
Method	A_N (↑)	F_N (↓)	A_N (↑)	F_N (↓)	A_N (↑)	F_N (↓)
Prompting (Centralized)	85.35	-	72.28	-	71.33	-
FedAvg-FT	10.23 ± 1.10	31.74 ± 0.80	12.03 ± 0.75	29.07 ± 0.66	18.76 ± 0.44	32.81 ± 1.22
FedLwF.MC [17]	59.08 ± 1.06	12.39 ± 0.76	52.87 ± 0.61	13.34 ± 0.38	62.39 ± 1.12	10.76 ± 0.50
FedAvg-Prompt	67.34 ± 1.42	8.38 ± 0.42	51.15 ± 0.68	8.84 ± 0.52	51.03 ± 2.23	12.03 ± 0.45
CFed [7]	72.26 ± 1.56	8.82 ± 0.64	45.64 ± 1.32	11.74 ± 1.22	63.32 ± 0.78	7.12 ± 0.66
TARGET [11]	73.56 ± 1.42	6.83 ± 0.91	52.38 ± 1.16	8.88 ± 0.96	61.84 ± 1.66	7.94 ± 0.52
HePCo (Ours)	$\mathbf{76.54 \pm 1.14}$	$\mathbf{6.61 \pm 0.73}$	$\mathbf{59.96 \pm 0.94}$	$\mathbf{7.08 \pm 0.40}$	$\mathbf{64.01 \pm 0.36}$	$\mathbf{6.83 \pm 0.31}$

First, to fine-tune the weights used for prompting, we pass the pseudo-data through the prompting mechanism of a model to obtain final prompts $\boldsymbol{p}$ which would serve as the target for distillation. Note, we do not require full models to generate these targets. Now to fine-tune the server model, we optimize for the Mean Squared Error (MSE) loss between the final prompts generated by the intermediate server model and each individual model (clients and last-task server); $\mathcal{L}_{prompt} = \sum_{c \in C} \mathcal{L}^{c}_{MSE} + \zeta^{y}_{t-1} L^{t-1}_{MSE}$, where $\mathcal{L}^{c}_{MSE}$ denotes the MSE loss between client c and the intermediate server model and L^{t-1}_{MSE} denotes the MSE loss between the intermediate model and the previous task server model. Further, ζ^{y}_{t-1} is an indicator variable which is set to 1 if y was seen in previous tasks and 0 if present in current task. Finally, to fine-tune the classifier of the server model, we minimize the cross entropy loss. The cross-entropy loss is computed between the predictions of the server for a batch of pseudo latents and the class labels that the pseudo latents were conditioned on.

5 Experiments

Setup. We appropriately adapt three image classification datasets commonly used in continual learning [14], to fit our specific setting. ImageNet-R and DomainNet capture real-world distribution shifts that can be challenging for models pre-trained on ImageNet to generalize to and are widely recognized benchmarks for evaluating continual learning in Foundation Models. We divide these datasets into 10-task (CIFAR-100, ImageNet-R) and 5-task (DomainNet) benchmarks. For all experiments reported in Table 1, we consider a class balanced setting ($\beta = 1$) and use a category ratio $\kappa = 0.6$ which means that if a task contains 10 categories, each active client is randomly assigned 6 of these categories. Further, we use a uniform split ratio $\gamma = 0.1$ which allows a client to be assigned 10% of the examples corresponding to the subset of categories. We evaluate all methods using the standard continual learning metrics of final average accuracy A_N and average forgetting F_N [15, 16].

5.1 Main Results

For fair comparison with existing SOTA methods, we adapt their implementations to use the same ViT backbone. We also report the performance of our *decomposed prompting* scheme in a centralized, traditional continual learning setting which can be thought of as an upper bound. The results presented in Tables 1 demonstrate the dominant performance of our method across all datasets and setups. The gains achieved by our method are more pronounced in the ImageNet-R setup which has longer task sequences and offer a significant shift from the pretrained distribution. All baselines that fine-tune the entire model are seen to struggle with longer sequences (CIFAR, Imagenet-R), showing significant forgetting. Our approach achieves absolute improvements of more than **7%** on ImageNet-R in average accuracy compared to TARGET [11], which is the current SOTA. Most importantly, HePCo achieves these solid results while enjoying low communication costs and without introducing any additional costs at the client-side. *We include further analysis in Appendix B.*

6 Conclusion

In conclusion, we propose HePCo (Heterogeneous Prompt Consolidation) for continual federated learning. Our method harnesses the prompt learning capabilities of foundation models to facilitate a data-free distillation framework for consolidating heterogeneous clients. We demonstrate the superior performance of our method through a series of experiments that emulate challenging real-world scenarios. By requiring clients to share parts of their models, we significantly reduce communication costs and enhance privacy. Importantly, our approach does not impose any additional overheads on the client side, making it highly valuable for real-world deployment.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 2239292

References

[1] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data, 2023.

[2] Yen-Chang Hsu, Yen-Cheng Liu, Anita Ramasamy, and Zsolt Kira. Re-evaluating continual learning scenarios: A categorization and case for strong baselines. *arXiv preprint arXiv:1810.12488*, 2018.

[3] Gido M van de Ven and Andreas S Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019.

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

[5] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data, 2020.

[6] Jaehong Yoon, Wonyong Jeong, Giwoong Lee, Eunho Yang, and Sung Ju Hwang. Federated continual learning with weighted inter-client transfer, 2021.

[7] Yuhang Ma, Zhongle Xie, Jue Wang, Ke Chen, and Lidan Shou. Continual federated learning based on knowledge distillation. In Lud De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 2182–2188. International Joint Conferences on Artificial Intelligence Organization, 7 2022. Main Track.

[8] Daiqing Qi, Handong Zhao, and Sheng Li. Better generative replay for continual federated learning, 2023.

[9] Nicholas Carlini, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramèr, Borja Balle, Daphne Ippolito, and Eric Wallace. Extracting training data from diffusion models, 2023.

[10] Jiahua Dong, Lixu Wang, Zhen Fang, Gan Sun, Shichao Xu, Xiao Wang, and Qi Zhu. Federated class-incremental learning, 2022.

[11] Jie Zhang, Chen Chen, Weiming Zhuang, and Lingjuan Lv. Addressing catastrophic forgetting in federated class-continual learning, 2023.

[12] Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. Learning imbalanced datasets with label-distribution-aware margin loss, 2019.

[13] Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning, 2022.

[14] James Seale Smith, Leonid Karlinsky, Vyshnavi Gutta, Paola Cascante-Bonilla, Donghyun Kim, Assaf Arbelle, Rameswar Panda, Rogerio Feris, and Zsolt Kira. Coda-prompt: Continual decomposed attention-based prompting for rehearsal-free continual learning. *arXiv preprint arXiv:2211.13218*, 2022. Accepted for publication at CVPR 2023.

[15] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-GEM. In *International Conference on Learning Representations*, 2019.

[16] Raphael Gontijo Lopes, Stefano Fenu, and Thad Starner. Data-free knowledge distillation for deep neural networks. *arXiv preprint arXiv:1710.07535*, 2017.

[17] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. *Advances in neural information processing systems*, 30, 2017.

[18] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019.

[19] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[20] Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, et al. Dualprompt: Complementary prompting for rehearsal-free continual learning. *arXiv preprint arXiv:2204.04799*, 2022.

[21] Daniel Justus, John Brennan, Stephen Bonner, and Andrew Stephen McGough. Predicting the computational cost of deep learning models, 2018.

[22] David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. Carbon emissions and large neural network training, 2021.

[23] Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. Quantifying the carbon emissions of machine learning, 2019.

Appendix

A Experimental Details

Implementation Details. For fair comparison, we use the ViT-B/16 backbone pretrained on Imagenet-1K as the encoder for all methods. We resize images to 224×224 and normalize to [0,1]. We implement our methods in PyTorch and use the PyTorch Image Models library [18] to obtain pretrained checkpoints. In our experiments, the total number of classes for CIFAR-100, ImageNet-R and DomainNet are 100, 200 and 345 respectively. We use 2 NVIDIA A40 GPUs for all experiments.

Training Details. For all methods, we use the Adam [19] optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ and train for 10 local epochs in each round. We learn each task through $R = 10$ communication rounds by selecting $C = 5$ stateless clients per round. Thus, we have 100 total rounds for a 10-task setup and 50 for a 5-task setup.

Hyperparameter Search. As done in DualPrompt [20] we use 20% of the training dataset as our validation data and conduct a hyperparameter search. We arrive at using a batch size of 64 for both local and server-side training. We use a learning rate of $1e^{-3}$ for our method and the prompting-based baselines and a learning rate of $5e^{-5}$ for all baselines that tune the entire model (FedAvg, FedLwF.MC). We search for learning rates in the values of $\{1e^{-6}, 5e^{-5}, 1e^{-5}, 5e^{-4}, 1e^{-4}, 5e^{-3}, 1e^{-3}, 5e^{-2}, 1e^{-2}\}$. For our method, we use a three-layer fully-connected network as our generator. We encode the class label using an embedding matrix of embedding length 64 and concatenate it with a noise vector of dimension 64. Our generator architecture can be described with having the following input sizes per layer : [128, 256, 1024] and an output size of 768 which is the dimension of the visual query. We train the generator for 100 epochs using a batch size of 64 and a learning rate of $1e^{-4}$ using the Adam optimizer. We fine-tune the server model using a learning rate of $1e^{-4}$ for 200 epochs. We use a *replay ratio* of 0.5 for our method, which means we mix 50 pseudo-latents corresponding to previous tasks for every 100 pseudo-latents corresponding to the current task. We conduct a search over values like [0, 0.125, 0.25, 0.375, 0.5, 0.625, 0.75, 0.875, 1] and find 0.5 to result into the best average accuracy A_N. We observe a *stability-plasticity* trade-off controlled by this hyperparameter with larger values leading to lower forgetting (F_N) but lower current task accuracies (*plasticity*) and smaller values yielding the opposite effect. Through the hyperparameter search we choose λ_{KL} and λ_{MSE} values to be 1 and 0.1 respectively.

B Ablation Studies and Additional Analysis

We perform ablations experiments on CIFAR-100 in the class-balanced setting from Table 1 and report in Table A

Ablating distillation of previous server model. By removing the previous task server model from the distillation and generation steps, we highlight its efficacy in alleviating forgetting. By ablating this component, we observe a significant drop in performance indicated by a rise in forgetting (F_N) and a drop in average accuracy (A_N). The underlying intuition is that without the replay of past task data, the method strongly prioritizes learning of the current task leading to a loss of knowledge from previously seen tasks.

Ablating disagreement losses in generation. To demonstrate the effectiveness of disagreement losses in generation, we set both the lambda coefficients to zero and observe a 6% drop. As discussed before, the intuition here is that in absence of the disagreement losses, the generator is prone to generate easily discriminable examples that lead to low classification loss but are less effective in distillation. To further highlight the importance of the individual losses, i.e $\mathcal{L}_{MSE}$ and $\mathcal{L}_{KL}$, we individually ablate them and observe performance drops.

Ablating distillation targets. In this experiment, we avoid distillation for the prompt components and the classifier separately and observe a decline in performance in both cases. The drop in performance is more pronounced when we ablate distillation for the classifier. This experiment highlights our decision to fine-tune both prompt components and classifiers by operating in the latent space.

Varying the category ratio. Figure A shows the performance of all methods for different values of *category ratio*. We observe that HePCo consistently outperforms competing methods without requir-

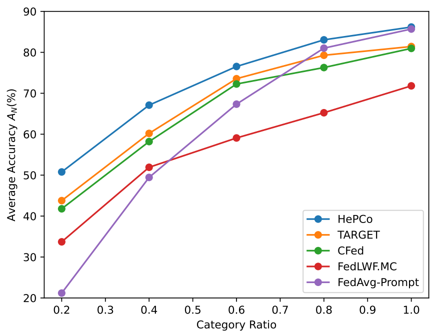

Figure A: Comparison of the methods under different category ratios

Method	A_N (↑)
HePCo (Ours)	$\mathbf{76.54 \pm 1.14}$
Ablate previous server model	61.15 ± 2.13
Ablate $\mathcal{L}_{KL}$ & $\mathcal{L}_{MSE}$	70.22 ± 1.45
Ablate $\mathcal{L}_{KL}$	71.39 ± 1.34
Ablate $\mathcal{L}_{MSE}$	74.11 ± 1.31
Ablate prompt distillation	74.42 ± 1.22
Ablate classifier distillation	68.46 ± 0.91

Table A: **Ablation Results (%) on 10-task CIFAR 100**. A_N gives the accuracy averaged over tasks and F_N gives the average forgetting.

ing any hyperparameter or design changes. The performance gap between HePCo and the competing methods widens with the category ratio, indicating its effectiveness in settings with high heterogeneity.

Imbalance experiments. In Table B, we benchmark the performance of all methods under two class imbalance setups characterized by $\beta = 0.05$ and $\beta = 0.01$. Our approach outperforms other methods across almost all setups by *even wider* margins compared to the class balanced setup of Table 1. This speaks volumes of the robustness of our method under conditions of extreme heterogeneity in comparison to existing SOTA approaches.

C Overhead Costs

Memory Overhead. Our method introduces additional parameters forming the prompting mechanism. The additional parameters amount to ~9.4% of the original size of the ViT encoder. Our method only needs to communicate the total learnable parameters in the model which includes the classifier and prompt components amounting to ~9.5% of the original model size. Methods that finetune the entire model need to learn and communicate all parameters in the encoder and classifier. Hence, our approach required only 9.5% of the communication costs compared to these approaches. Furthermore, the current state-of-the-art methods like CFed and TARGET require communicating a dataset of images (obtained from the surrogate dataset or a generative mechanism) after every round or task which significantly increases the communication overhead in addition to sharing complete models!

Computation Overhead. Our method does not require any extra computation at the client side but introduces an overhead at the server side. This overhead includes the time required to train the generators and perform knowledge distillation. To quantify this overhead, we conducted benchmarking using 2 NVIDIA TITAN RTX GPUs in a 5 client setup, as described in the experiments section. Our method adds an extra 220 seconds of computational time at the server side per round, in contrast to the 166 seconds introduced by CFed and the 190 seconds incurred by TARGET. It is crucial to emphasize that our method imposes no additional overhead on the client side, unlike CFed and TARGET, where the client is effectively responsible for learning the current task and distilling knowledge from past tasks. In most practical federated learning scenarios, edge devices have limited computational capacity compared to the server. Our approach prioritizes client-level efficiency, even if it entails a slight trade-off in server-level efficiency.

Storage Overhead. As our method operates in a stateless FL setup, we do not require clients to maintain any state information or additional storage. Our approach requires the server model to store the classifier and prompt components corresponding to the last task model which is used in distillation resulting into a storage cost equal to ~9.5% of the base encoder model size. Other baselines [17] incur extra storage costs at the client side equal to the size of entire encoder and classifier i.e ~86M parameters. Additionally, CFed and TARGET incur costs equivalent to storing an entire image dataset at both server and individual client levels.

In summary, our approach attains state-of-the-art performance while imposing lower overheads compared to existing methods.

Table B: **Results (%)** for class-imbalanced setup

Datasets	CIFAR-100		ImageNet-R		DomainNet	
Method	A_N (↑)		A_N (↑)		A_N (↑)	
Imbalance ratio (β)	$\beta = 0.05$	$\beta = 0.01$	$\beta = 0.05$	$\beta = 0.01$	$\beta = 0.05$	$\beta = 0.01$
FedAvg-FT	8.81 ± 1.53	9.18 ± 1.26	9.26 ± 1.02	8.88 ± 1.24	13.02 ± 1.29	11.65 ± 1.84
FedLwF.MC [17]	50.40 ± 0.88	40.39 ± 1.06	19.94 ± 0.78	13.34 ± 1.41	57.34 ± 0.84	52.46 ± 0.72
FedAvg-Prompt	62.72 ± 1.79	54.43 ± 1.57	36.51 ± 0.86	28.16 ± 1.12	47.73 ± 1.25	43.23 ± 1.03
CFed [7]	70.26 ± 1.20	$\mathbf{62.04 \pm 1.62}$	34.62 ± 1.41	25.74 ± 1.08	59.89 ± 0.68	55.22 ± 0.80
TARGET [11]	66.47 ± 1.22	58.13 ± 1.54	30.20 ± 1.35	19.84 ± 1.41	56.44 ± 0.45	51.82 ± 0.58
HePCo (Ours)	$\mathbf{70.34 \pm 1.08}$	61.70 ± 1.48	$\mathbf{45.45 \pm 0.98}$	$\mathbf{41.68 \pm 1.44}$	$\mathbf{61.10 \pm 0.76}$	$\mathbf{58.82 \pm 0.84}$

D Discussion

Limitations. It is worth noting that prompting-based methods are still relatively new and not extensively studied, making the interpretation of these prompts challenging. Therefore, future work should focus on testing the robustness of these methods in diverse setups to ensure their effectiveness in different scenarios. One potential limitation of this work is in the computation overhead introduced at the server, which may be an issue for some use-cases. Although the generation and distillation procedures are relatively lightweight, they still rely on server-side compute resources, which may not be universally accessible in all scenarios. Additionally, our approach necessitates clients to use pretrained vision transformers, leaving open the question of how this framework can be extended to accommodate other architectures. These are interesting avenues for future research.

Broader Impact. The machine learning community is increasingly leaning towards the adoption of large-scale models for various applications. However, updating these models with new data poses a significant challenge. Retraining models from scratch each time new data arrives is computationally expensive and can have substantial financial [21] and environmental [22, 23] implications. Our approach offers a solution by enabling incremental learning on new data without the need for complete model retraining. Additionally, our use of prompting techniques allows for significant reductions in communication and local computation costs while enhancing privacy, which is especially critical for on-device edge computing applications.