Augmenting Federated Learning with Pretrained Transformers

Anonymous Author(s)
Affiliation
Address
email

Abstract

The explosive growth and diversity of machine learning applications motivate a fundamental rethinking of learning with mobile and edge devices. How can we address *diverse/disparate client goals* and learn with *scarce heterogeneous data*? While federated learning (FL) aims to address these issues, it has several bottlenecks and challenges hindering a unified solution. On the other hand, large transformer models have been shown to work across a variety of tasks often achieving remarkable few-shot adaptation. This raises the question: Can FL clients use a single general-purpose model – rather than custom models for each task – while obeying *device and network constraints*? In this work, we investigate pretrained transformers (PTF) to achieve these on-device learning goals and thoroughly explore the roles of model size and modularity, where the latter refers to adaptation through modules such as prompts or adapters. We demonstrate that: **(1) Larger scale** shrinks the accuracy gaps between alternative approaches and improves heterogeneity robustness. Crucially, scale allows clients to run *more local SGD epochs* which substantially ($\times 4$) reduces the number of communication rounds. At the extreme, clients can achieve respectable accuracy fully-locally reducing the need for collaboration. **(2) Modularity** enables $>100\times$ less communication in bits. Surprisingly, it also boosts the generalization capability of local adaptation methods and the robustness of smaller PTFs. To explain these benefits, we show that scale and modularity can synergistically mitigate the *representation shift* during FL. Finally, to harness multitasking capabilities of modern PTFs, we propose FedYolo: A new FL approach that assigns both dedicated and shared modules to FL tasks to manage their interference. Our extensive experiments demonstrate FedYolo's value and the power of scale and modularity for multitasking.

1 Introduction

Federated learning (FL) has enjoyed significant success in enabling collaboration across large number of decentralized clients. Nevertheless, FL confronts challenges due to the limited client data, the heterogeneous nature of FL scenarios, and the necessity for multitasking, all of which can lead to issues like catastrophic forgetting(e.g. when client updates override each other)[21, 9, 19]. Despite rich FL literature, we still lack a clear unified strategy that overcomes these challenges. Meanwhile, PTFs can be few-shot *adapted* to various downstream tasks(i.e. **power of scale** [6, 20]), providing a warm-start for FL and better adaptation to local client distributions. Advances in mobile hardware[17] and model compression/distillation [14, 29, 32] enable the deployment of smaller, equally effective models on clients' devices.

However, it remains uncertain whether these benefits can be realized in multitask FL setting that involves heterogeneous data and communication bottlenecks. In this work, together with **scale**, we identify the **power of modularity** to address FL-specific challenges. The training strategies and

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

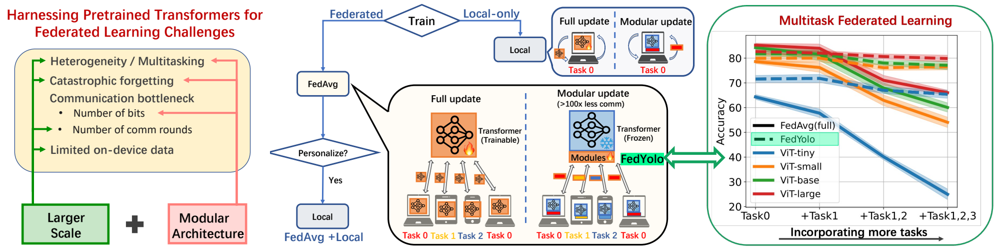


Figure 1: **Left side:** We investigate *scale and modularity* of pretrained transformers (PTF) to address federated learning (FL) challenges. **Center:** The training can branch into either FL or `Local-only` learning, once PTF model is loaded to the device. FL uses either `FedAvg` or `FedAvg+Local`. All three training schemes could be implemented with two update methods: Full-update and modular-update as shown in the center box. Full-update trains and communicates all model parameters whereas modular-update trains a small subset of parameters while freezing the PTF backbone. **Right side:** For multitask FL, we propose `FedYolo` which assigns unique modules for each task (distinct colors of Tasks A,B,C). `FedYolo` is superior to `FedAvg` (with full-update) as number of tasks grow thanks to modularity.

update methods we explore are depicted in Figure 1. Updating only modules significantly reduces communication costs, as shown in supplementary materials. To explore benefits of PTFs, we study three training schemes, `Local-only` learning, `FedAvg`, and `FedAvg+Local`, for FL settings with heterogeneous data across tasks. In a nutshell, our main message is:

Large PTFs with modular updates naturally enable communication-efficient, robust, multitask FL.

This message generalizes well across different module choices (prompt, LoRA, adapter), pointing to the universal benefit of parameter-efficient FL. Specifically, we make the following contributions:
• **Need for collaboration / personalization.** Scale allows for better few-shot learning and reduces the reliance on personalization and collaboration by shrinking the accuracy gaps between `FedAvg` and `FedAvg+Local` as well as `FedAvg+Local` and `Local-only` learning. We also found that modular-update often outperforms full-update under few-shot or heterogeneous data. This makes modular-update a surprisingly effective strategy for `Local-only` learning and `FedAvg+Local`. Importantly, combined benefits of modularity and scale make `Local-only` learning fairly competitive with FL.

• **Heterogeneity, Local SGD, Communication.** We find that scale boosts robustness of FL to data heterogeneity, while the modularity particularly improves the robustness of smaller PTFs. They also both provide resilience to forgetting: Accuracy of `FedAvg+Local` remains competitive with `FedAvg` on the global distribution even after the local-learning phase. In synergy, *larger scale significantly reduces the number of communication rounds* by allowing clients to run much more local SGD epochs ($\times 4$ in Fig 7) without sacrificing global accuracy. We provide theoretical insights into these by demonstrating large model incur small *representation shift* even when trained with many epochs. Modules have in the order of tens of thousands of parameters, thus, modular updates unlock orders-of-magnitude communication savings compared to full update. We find that, this occurs while maintaining, and *often accelerating, the rate of convergence* in communication rounds.

• **Multitask learning.** In a multitask setting where FL clients collaboratively and simultaneously learn multiple disparate tasks (e.g., classification on different domains such as CIFAR, CelebA, and FEMNIST datasets), the challenge is determining which parts of the model to update. Building on modularity and *"one PTF for many tasks"*, we propose the `FedYolo` algorithm ("You Only Load Once") that assigns isolated modules to each task while keeping the PTF backend frozen. Fig. 1 (right side) demonstrates that `FedYolo` performs on par with learning each task in isolation whereas multitasking with full-update suffers from catastrophic forgetting even for large PTFs.

Our findings have important implications. Adapting large PTFs via modular-update not only provides a simple communication-efficient strategy with relatively minor drawbacks but also provides significant potential upsides in terms of personalization, robustness, and multitasking. Notably, scale and modularity makes `Local-only` learning a fairly competitive alternative to FL approaches `FedAvg` or `FedAvg+Local`, hinting at the viability of full privacy on the client side. Additionally, our proposal `FedYolo` enables the clients to use a single PTF and multiple small modules to address diverse set of mobile ML goals, avoiding the need for maintaining/training multiple models.

2 Related Work

Federated Learning. Data heterogeneity, multitasking, and personalization have been studied in FL in various settings [21, 7, 23, 22]. Much of the prior works focus on the design of algorithms

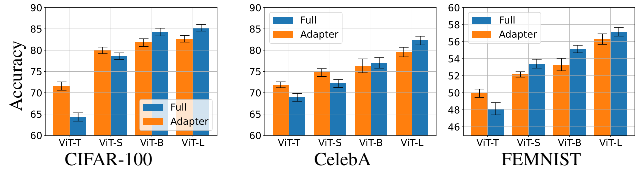
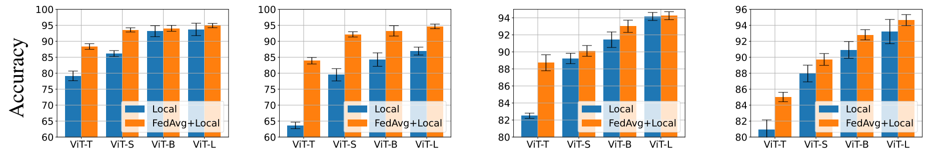


Figure 2: Model performance of `FedAvg` with hetereogeneous data distribution. Larger PTFs outperform smaller PTFs.



(a) CIFAR-100, modular-update

(b) CIFAR-100, full-update

(c) CelebA, modular-update

(d) CelebA, full-update

Figure 3: Performance comparison between `Local-only` learning (blue) and `FedAvg+Local` (orange), for different datsets and update strategies (full-update and modular-update). As PTF scale increases, local training become more comparable to the federated setting.

rather than the model architecture. Closer to our work, [28] proposed FL with pretrained models, however only consider full updates whereas modularity is central to our message. Recent works [35, 36] explore related modular-update ideas for federated learning, however don't explore the role of scale. Importantly, ours is the only work that explores and provides a concrete solution for multitask learning with PTFs.

Parameter-efficient tuning and pretrained transformers. PTFs have garnered significant attention in machine learning, owing to their impressive performance across a wide variety of applications[2, 5]. Although non-federated, recent works explored the benefits of scale (model size as well as data and computation during pretraining) in robustness to forgetting [26] and (few-shot) accuracy [10, 33, 11]. Parameter-efficient tuning methods have shown significant promise for enabling lightweight adaptation of transformer architectures.

3 Experiments

Preliminaries and Experimental Setup: Following [25], we evaluate the performance on CIFAR [18] and two real-world datasets CelebA and FEMNIST [3] from the LEAF benchmark [3], following [25, 26]. For CIFAR, we simulate three data partitions("homogeneous", "mild heterogeneous" and "more heterogeneous") and control the non-IID level by changing the number of classes included in each client. Importantly, all our experiments focus on the few-shot setting where we train on subsets of these datasets. For instance, our CelebA and FEMNIST experiments use 2.6% and 1.8% fraction of the total sample size respectively. Due to space limitations, we only include the results of the Adapter method in the main paper, while the results of the LoRA and VPT methods are similar and relegated to the Appendix. For evaluation metrics, unless otherwise stated, the evaluation of models is the average local accuracy across clients. Further details are in the supplementary material D.

3.1 PTF Scale Boosts Performance

Larger PTFs improve model performance: The impact of scale in FL is an underexplored topic. To evaluate performance on a heterogeneous data distribution, we use both simulated and real-world data heterogeneity. In Figure 4, the simulated data heterogeneity setting involves clients with different class distributions. In Figure 2, the real-world data heterogeneity involves clients with both different class distributions and different domains, such as each client having data relating to a particular celebrity in CelebA. In all cases, larger PTFs outperform smaller PTFs.

Larger PTFs narrow the local vs. federated training gap: Intuitively, federated learning should perform better since information is shared between clients, but larger PTFs may approach the performance of federated learning. In Fig. 3, we compare the performance of `Local-only` learning and `FedAvg+Local` for the full-update and modular-update training strategies. The results show that `Local-only` learning becomes increasingly competitive with `FedAvg+Local` as the model scale increases. Moreover, employing modules can help achieve better performance and narrow the gap between fully local and federated training (the gap between `Local-only` learning and `FedAvg+Local`

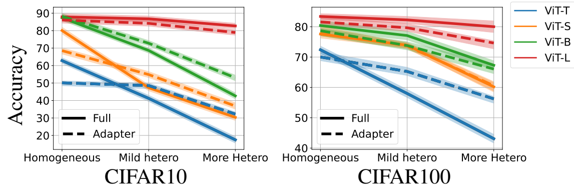


Figure 4: Test set accuracy under different levels of client data heterogeneity. Larger PTFs show consistently better performance, especially in more heterogeneous settings. Comparing the proportion of the same curve's descent from left to right, we observe that larger scale and modularity can enhance performance.

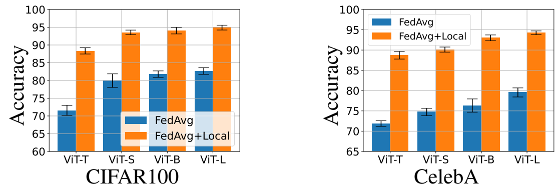


Figure 5: Test set accuracy with (`FedAvg+Local`) and without (`FedAvg`) personalization of modular-update. As the PTF scale increases, the gap between the two approaches diminishes. full-update results are shown in supplementary materials.

is smaller in Fig. 3a than Fig. 3b). In other words, if clients wish to avoid federated learning, large PTFs with modular updates that are trained on-device can achieve reasonable performance.

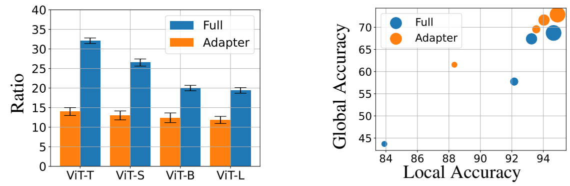
Modular updates can outperform full updates: In a previous study [20], it was demonstrated that with larger scale PTFs, the performance gap between full updates and prompt tuning could be reduced. However, federated learning introduces additional challenges related to heterogeneity and decoupled data, leading to more interesting findings. In particular, we find that modular approaches can actually outperform full updates in certain situations. With heterogeneous data distribution in Fig. 2, the ViT-T PTF sometimes have higher accuracy with the modular-update. The advantage of modular-update is more pronounced when the data is even more heterogeneous, as depicted on the right half of Fig. 4. We conclude that full-update is more susceptible to issues introduced by federated learning, especially when using small-scale PTFs, and modular approaches can sometimes outperform them.

3.2 Heterogeneous Client Data Distributions

Enhancing Robustness to Heterogeneous Distributions: In Fig. 4, we plot the accuracy for full-update and modular-update training strategies, for varying amounts of data heterogeneity on the clients. The results show a notable decrease in test accuracy on heterogeneous data partitions when training smaller PTFs with full updates (solid blue curve), particularly in the highly heterogeneous setting. Employing larger PTFs or modular update maintains accuracy even under significant heterogeneity. Larger PTFs consistently outperform, irrespective of heterogeneity level or fine-tuning method. If PTFs are not sufficiently large, performance plummets as heterogeneity escalates (e.g., the solid blue curve). In contrast, modular update can enhance performance.

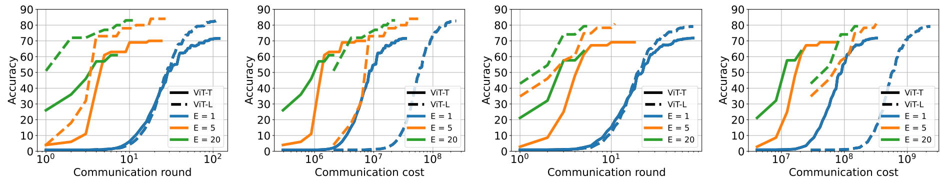
Bridging the Personalization Gap: We next explore whether PTFs and modularity can help reduce the disparity between personalized training and the average global model. As shown in Figure 5, the disparity shrinks as the scale of the PTFs grows, for different datasets and update strategies. Full update tends to widen this gap, especially with smaller backbones, in contrast to the modular update. This suggests that employing larger PTFs and modular update could mitigate the necessity for computationally intensive personalized training.

Mitigating Catastrophic Forgetting: We examine if larger scale and modularity can alleviate catastrophic forgetting, as depicted in Fig. 6. The model's performance is compared pre- and post-personalization to induce forgetting. Initially, the model is trained on a global dataset, followed by personalization by training on a client-specific local dataset with fewer classes. Maintaining performance on the full set of 100 classes alongside improving accuracy on the local classes indicates better forgetting resistance. Fig. 6a shows the forgetting ratio ($\frac{Acc_{\texttt{FedAvg}} - Acc_{\texttt{FedAvg+Local}}}{Acc_{\texttt{FedAvg}}}$, smaller is better). The results demonstrate that modular-update significantly reduces the forgetting ratio, with this ratio decreasing as the PTF scale increases. Fig. 6b plots the global vs. local accuracy. The results show that larger PTFs and modularity enable personalized models to simultaneously achieve higher global and local accuracy, effectively mitigating catastrophic forgetting.



(a) Forgetting ratio (b) Personalized model accuracy

Figure 6: (a) Global accuracy forgetting ratio vs scale and tuning methods. modular-update and larger scale mitigates catastrophic forgetting. (b) `FedAvg+Local` accuracy on local (new distribution, 20 personalized classes) and global (previous distribution, 100 classes) test sets. Values towards the upper right corner are better. modular-update with large PTFs exhibit better retention of information from the previous class distribution.



(a) CIFAR100 (b) CIFAR100 (c) CelebA (d) CelebA

Figure 7: We conducted experiments comparing two different scales of PTFs, ViT-L and ViT-T. The color indicates the number of local training epochs (E). All experiments used the modular-update. To highlight convergence speed, we employed early stopping when the training reached convergence.

3.3 Reducing Federated Communication Cost

We aim to reduce communication cost while preserving accuracy, an objective intuitively achieved through modules that require fewer parameters for training than updating the entire model. When we compare the communication rounds and communication costs between the modular-update and full-update approaches, both with a default of one local epoch, we observe two significant benefits: **Modularity decreases communication rounds** and **Modularity significantly reduces communication cost, by over 100**$\times$. The details are shown in C.2

Large PTFs allow more local epochs: Large local training epochs (E) can reduce communication costs. However, a larger E may result in a decline in final performance on heterogeneous data partitions. Our study demonstrates that larger scales of PTFs can enable larger local training epochs even with heterogeneous data partitions. The results are presented in Fig. 7.

Fig. 7a shows that using larger local training epochs (E) can significantly accelerate convergence. For fine-tuning with small PTFs, it was observed that larger E truly negatively impacted the performance. However, larger-scale PTFs can maintain or even improve performance when larger E values are used. We also compare the communication cost. As shown in Figure 13, larger-scale PTFs generally exhibit a higher communication cost due to larger number of trainable parameters. However, our findings in Figure 7b reveal that by simply using larger local epochs (E), larger-scale PTFs can achieve comparable or even better performance than smaller-scale PTFs within a fixed communication cost budget. For instance, for CIFAR100 dataset, when the communication cost is limited to 10^7, ViT-T($E = 1$) achieves an accuracy of 54.61, while ViT-L($E = 20$) achieves 75.04.

3.4 Representation-based Explanation of Power of Scale

To shed light on our findings, we propose a representation-theoretic explanation. When fine-tuning the model for new tasks, larger PTFs tend to undergo less dramatic alterations in their feature embeddings. This concept is depicted in Figure 8. Here, we employ a bubble analogy to represent the model's representational strength, where larger models/bubbles symbolize richer features. The expansive representation capacity of large PTFs allows them to encapsulate a wide spectrum of features that are inherently adaptable across diverse tasks. Notably, larger pre-trained models necessitate minor adjustments to adapt to Task 1. This would imply smaller changes in feature embeddings of Task 1 itself as well as an external Task 2 (which is not used in fine-tuning). We provide empirical justification for this hypothesis through the experiments provided in Sec. C.3

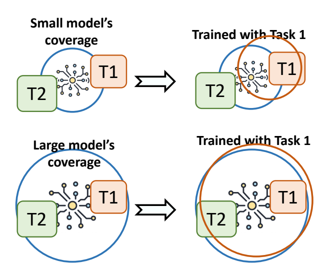


Figure 8: We fine-tune a model (pre-trained on ImageNet-21k) on a specific task, Task 1. An additional Task 2, not involved in training, is also shown. We depict the *representation shift* (blue→red bubbles) as the model is tuned. Larger transformer has better coverage and robust to shift.

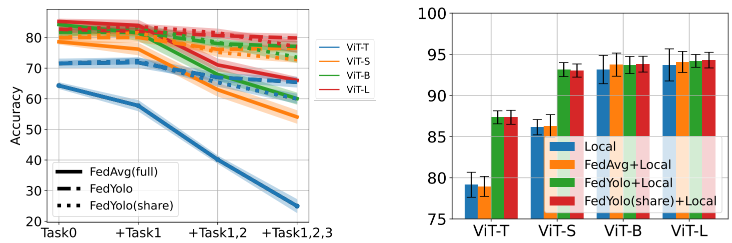


Figure 9: Task 0 (CIFAR100) performance of multi-task federated learning as we incorporate more tasks to the problem (y-axis). (L) Global accuracy, (R) Local accuracy with adaptation. In either scenario, `FedYolo`, with its task-specific modules, outperforms conventional `FedAvg`.

4 Modular Multitask Learning with FedYolo

Task	Dataset	# clients	# samples	Data partition
0	CIFAR100	20	100	mild hetero
1	CIFAR10	20	100	mild hetero
2	CelebA	787	$\leq$8	celebrity identity
3	FEMNIST	532	$\leq$120	character writer

Table 1: Details of data partitioning for multi-task learning

Traditional `FedAvg` entails high communication costs and vulnerability to heterogeneity due to shared full PTF parameters. The experiments within Section 3 have demonstrated the potential of large-scale PTFs and modularity to reduce communication costs and boost robustness, making them promising for multi-task federated learning. Based on these findings, we propose `FedYolo` as a multi-task federated learning method, as illustrated in Figure 1. `FedYolo` assigns both shared-across-tasks and task-dedicated modules and all modules are plugged into a single frozen PTF. This PTF is loaded once at the start of the training, equipping clients with a backbone architecture. The task-specific modules are then updated and communicated with minimal cost going forward. In the vanilla version, each client trains and sends the modules for their own tasks. This might potentially suffer from privacy leak as the server will know which client has what task/module. An alternative is letting clients send modules for all tasks, where most entries are zero and only the tasks at hand have non-zero entries. Combined with secure aggregation techniques [8, 24], this will ensure that the server will not learn which clients contributed to a particular task. The detailed algorithm is in the supplemental materials.

To evaluate `FedYolo`, we train clients on multiple tasks simultaneously, including image classification on CIFAR-10, CIFAR-100, CelebA, and FEMNIST datasets, where each client is assigned to one task. The task assignments and data partitioning details are in Table 1. A `FedAvg` baseline with full-update is also trained on the same tasks. We display the evolving accuracy of Task 0. The results in Figure 9 show `FedYolo` (dashed line) consistently outperforming conventional `FedAvg` (solid line), particularly with more tasks and for smaller PTFs. To assess the impact of module sharing, we also compare `FedYolo`, without module sharing across tasks, with `FedYolo`(share), where tasks share the modules in initial layers. When introducing a related task (Task 1, CIFAR10), `FedYolo`(share) benefits from multitasking, for instance, for ViT-L, `FedYolo`(share) demonstrates a 1.7% performance improvement compared to `FedYolo`. Conversely, when incorporating unrelated tasks (Task 2, 3), `FedYolo`(share) slightly degrades compared to `FedYolo` but is still significantly more robust than `FedAvg` and mostly maintains Task 0's accuracy.

To examine the impact of personalization, we conduct another experiment where we add local training for clients after the federated training is complete. The results in Fig. 9(right) show that `FedYolo` also surpasses `FedAvg` and `Local-only` learning with personalized models in terms of accuracy, especially with smaller models. The performance gap narrows with larger models, supporting larger PTFs' role in balancing local and federated training. With larger PTFs, users can exclusively train locally with similar performance, valuable where data privacy is vital. Across PTF sizes, the near-identical performance of `Local-only` learning and `FedAvg+Local` implies standard `FedAvg`'s limited impact on the global model's generalization ability, whereas `FedYolo` provides clear improvements by avoiding interference across distinct tasks.

References

[1] Durmus Alp Emre Acar, Yue Zhao, Ramon Matas Navarro, Matthew Mattina, Paul N Whatmough, and Venkatesh Saligrama. Federated learning based on dynamic regularization. *arXiv preprint arXiv:2111.04263*, 2021.

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[3] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018.

[4] Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models. *arXiv preprint arXiv:2203.06904*, 2022.

[5] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021.

[7] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. *Advances in Neural Information Processing Systems*, 33:3557–3568, 2020.

[8] Hossein Fereidooni, Samuel Marchal, Markus Miettinen, Azalia Mirhoseini, Helen Möllering, Thien Duc Nguyen, Phillip Rieger, Ahmad-Reza Sadeghi, Thomas Schneider, Hossein Yalame, et al. Safelearn: Secure aggregation for private federated learning. In *2021 IEEE Security and Privacy Workshops (SPW)*, pages 56–62. IEEE, 2021.

[9] Dashan Gao, Xin Yao, and Qiang Yang. A survey on heterogeneous federated learning. *arXiv preprint arXiv:2210.04505*, 2022.

[10] Tom Henighan, Jared Kaplan, Mor Katz, Mark Chen, Christopher Hesse, Jacob Jackson, Heewoo Jun, Tom B Brown, Prafulla Dhariwal, Scott Gray, et al. Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*, 2020.

[11] Danny Hernandez, Jared Kaplan, Tom Henighan, and Sam McCandlish. Scaling laws for transfer. *arXiv preprint arXiv:2102.01293*, 2021.

[12] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp, 2019.

[13] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021.

[14] Yating Huang, Yunzhe Hao, Jiaming Xu, and Bo Xu. Compressing speaker extraction model with ultra-low precision quantization and knowledge distillation. *Neural Networks*, 154:13–21, 2022.

[15] Yufei Huang, Yujia Qin, Huadong Wang, Yichun Yin, Maosong Sun, Zhiyuan Liu, and Qun Liu. Fpt: Improving prompt tuning efficiency via progressive training. *arXiv preprint arXiv:2211.06840*, 2022.

[16] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning, 2022.

[17] Bertalan Kovács, Anders D Henriksen, Jonathan Dyssel Stets, and Lazaros Nalpantidis. Object detection on tpu accelerated embedded devices. In *Computer Vision Systems: 13th International Conference, ICVS 2021, Virtual Event, September 22-24, 2021, Proceedings 13*, pages 82–92. Springer, 2021.

[18] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[19] Priyush Kumar, Indrajeet Kumar Sinha, and Krishna Pratap Singh. Mtfl: Multi-task federated learning for classification of healthcare x-ray images. In *Computer Vision and Image Processing: 7th International Conference, CVIP 2022, Nagpur, India, November 4–6, 2022, Revised Selected Papers, Part II*, pages 572–585. Springer, 2023.

[20] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

[21] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE signal processing magazine*, 37(3):50–60, 2020.

[22] Xin-Chun Li and De-Chuan Zhan. Fedrs: Federated learning with restricted softmax for label distribution non-iid data. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 995–1005, 2021.

[23] Wei Yang Bryan Lim, Nguyen Cong Luong, Dinh Thai Hoang, Yutao Jiao, Ying-Chang Liang, Qiang Yang, Dusit Niyato, and Chunyan Miao. Federated learning in mobile edge networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 22(3):2031–2063, 2020.

[24] Mohamad Mansouri, Melek Onen, Wafa Ben Jaballah, and Mauro Conti. Sok: Secure aggregation based on cryptographic schemes for federated learning. *Proc. Priv. Enhancing Technol*, (1):140–157, 2023.

[25] Liangqiong Qu, Yuyin Zhou, Paul Pu Liang, Yingda Xia, Feifei Wang, Ehsan Adeli, Li Fei-Fei, and Daniel Rubin. Rethinking architecture design for tackling data heterogeneity in federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10061–10071, 2022.

[26] Vinay Venkatesh Ramasesh, Aitor Lewkowycz, and Ethan Dyer. Effect of scale on catastrophic forgetting in neural networks. In *International Conference on Learning Representations*, 2022.

[27] Andreas Steiner, Alexander Kolesnikov, , Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, and Lucas Beyer. How to train your vit? data, augmentation, and regularization in vision transformers. *arXiv preprint arXiv:2106.10270*, 2021.

[28] Yue Tan, Guodong Long, Jie Ma, Lu Liu, Tianyi Zhou, and Jing Jiang. Federated learning from pre-trained models: A contrastive learning approach. *arXiv preprint arXiv:2209.10083*, 2022.

[29] Chaofan Tao, Lu Hou, Wei Zhang, Lifeng Shang, Xin Jiang, Qun Liu, Ping Luo, and Ngai Wong. Compression of generative pre-trained language models via quantization. *arXiv preprint arXiv:2203.10705*, 2022.

[30] Orion Weller, Marc Marone, Vladimir Braverman, Dawn Lawrie, and Benjamin Van Durme. Pretrained models for multilingual federated learning. *arXiv preprint arXiv:2206.02291*, 2022.

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

[32] Canwen Xu and Julian McAuley. A survey on model compression for natural language processing. *arXiv preprint arXiv:2202.07105*, 2022.

[33] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12104–12113, 2022.

[34] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

[35] Zhuo Zhang, Yuanhang Yang, Yong Dai, Lizhen Qu, and Zenglin Xu. When federated learning meets pre-trained language models' parameter-efficient tuning methods. *arXiv preprint arXiv:2212.10025*, 2022.

[36] Haodong Zhao, Wei Du, Fangqi Li, Peixuan Li, and Gongshen Liu. Reduce communication costs and preserve privacy: Prompt tuning method in federated learning. *arXiv preprint arXiv:2208.12268*, 2022.

A Organization of the Appendix

In Section B, we provide a detailed supplementary explanation of Section 4. We have included the algorithm box for `FedYolo` and conducted further research on the impact of shared-across-tasks modules.

In Section C, we add additional experiments. Specifically, we make the following observations:

- Large PTFs allow for using more local epochs without sacrificing accuracy. This reduces the number of communication rounds in federated learning.
- We provide evaluations for FedProx which is a state-of-the-art optimization-based federated learning method for heterogeneity. In line with main submission, `FedYolo` outperforms FedProx with full-updates in multitask settings.
- In main submission, we only compared full-update and modular-update. An alternative is only tuning the classifier head i.e. the final layer(s). We find that modular-update achieves superior performance compared to only head-tuning under similar number of trainable parameters.
- Our main empirical findings generalize well across module types (LoRa, Adapted, prompt-tuning).
- Larger PTF retain its benefits over smaller PTF even if we use the same module size (i.e. equalizing the number of trainable parameters). We conducted this experiment because in the main body of the paper, we used the default module sizes which are proportional to the embedding dimension, thus, larger PTFs were using larger modules.

In Section D, we provide further experiment details.

B Further discussion of FedYolo

Our `FedYolo` method is described in Algorithm 1. The trainable parameters could contain both shared-across-tasks and task-dedicated modules. The vanilla `FedYolo` assigns a unique module to each task, a distinct module is allocated to each task, preventing mutual benefit or detriment among tasks. This leads us to question whether it's feasible to leverage the advantages of task sharing while circumventing vulnerability to heterogeneity. This can potentially be achieved by integrating the robustness of large-scale architecture and modularity. Consequently, our `FedYolo` also incorporates shared-across-task modules.

To investigate the impact of task sharing, we conducted experiments with two sharing options: sharing the modules in initial models (half) or sharing all modules. The results are illustrated in Fig.10. When Introducing related tasks, `FedYolo` (share) consistently yields multitasking benefits. On the other hand, when integrating an unrelated task, `FedYolo` (share) experiences a minor reduction in performance compared to `FedYolo`. However, it maintains significantly higher robustness than `FedAvg` and predominantly preserves Task 0's accuracy. Remarkably, even without computationally demanding methods such as Neural Architecture Search (NAS), the shared modules can be conveniently selected, resulting in comparable performance. Figure 10c further illustrates that when both `FedAvg` and `FedYolo` share all trainable parameters among tasks, `FedYolo` still exhibits substantial enhancement. Thus, the advantage of `FedYolo` stems not only from assigning distinct modules to each task to reduce the impact of task heterogeneity but also from effectively leveraging the robustness inherent in large pretrained transformers and modularity.

C Additional Experiments

C.1 Homogeneous client data distributions

Larger PTFs improve model performance: In a federated setting, the presence of heterogeneous data distributions and limited samples among clients can lead to challenges in achieving optimal performance. The generalization benefits of large models (a.k.a. over-parameterization) has been explored empirically as well as theoretically [34]. However, the impact of scale in FL is an under-explored topic and it is not immediately clear whether large models will retain their benefits in FL

Algorithm 1 `FedYolo`

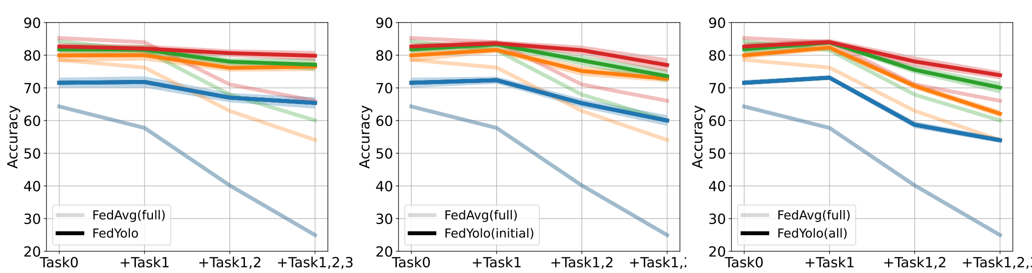
Parameters: Client set $\mathcal{C}$; # of rounds T; # of local epochs E; # of tasks K; # of clients per round M;
PTF parameters $\mathcal{W}_{\texttt{frozen}}$; trainable parameters $\mathcal{W}^{k}_{\texttt{train}}$ for task k (containing task-specific head and module);
Local dataset $\mathcal{D}_m$ of client m.

Load and freeze PTF $\mathcal{W}_{\texttt{frozen}}$ on each client
for each communication round $t = 1$ to T **do**
  $\mathcal{C}^t \leftarrow$ (randomly sample M clients from $\mathcal{C}$)
  for each client $m \in \mathcal{C}^t$ **in parallel do**
    $k \leftarrow$ task ID of client m
    Load $\mathcal{W}^{t,k}_{\texttt{train}}$ to the client
    $\mathcal{W}^{t+1,m,k}_{\texttt{train}} \leftarrow \textsc{LocalTuning}(m, \mathcal{W}^{t,k}_{\texttt{train}})$
    Send client parameters $\mathcal{W}^{t+1,m,k}_{\texttt{train}}$ to server
  end for
  for task $k = 1$ to K **do**
    $\mathcal{C}^{t,k} \leftarrow$ clients in $\mathcal{C}^t$ with task k
    $\mathcal{W}^{t+1,k}_{\texttt{train}} \leftarrow \texttt{Average}(\{\mathcal{W}^{t+1,m,k}_{\texttt{train}}\}_{m \in \mathcal{C}^{t,k}})$
  end for
end for

function $\textsc{LocalTuning}(m, \mathcal{W}^{t,k}_{\texttt{train}})$
  $\mathcal{W}^t \leftarrow$ (assemble task-specific $\mathcal{W}^{t,k}_{\texttt{train}}$ and $\mathcal{W}_{\texttt{frozen}}$)
  for epoch $e = 1$ to E **do**
    $\mathcal{W}^{t+1,m,k}_{\texttt{train}} \leftarrow$ train $\mathcal{W}^{t,k}_{\texttt{train}}$ on dataset $\mathcal{D}_m$
  end for
  Send $\mathcal{W}^{t+1,m,k}_{\texttt{train}}$ to the server
end function

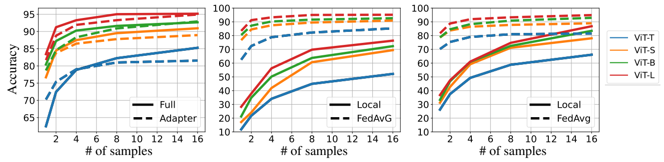
setting with local training and limited samples. To study this, we conduct experiments in a federated few-shot setting, exploring both homogeneous and heterogeneous data distributions to understand the effects on model performance. For a homogeneous data distribution where all clients have the complete set of 100 classes from CIFAR-100, we plot the accuracy as the number of samples per class increases in Fig. 11a. We can see that larger PTFs consistently outperform smaller PTFs, regardless of whether the $full - update$ or $modular - update$ method is employed.

Larger PTFs narrow the local vs. federated training gap: The good performance of large PTFs raises the question of whether it is preferable to simply have clients store large PTFs locally and train them, without joint training through federated learning. To study this, we conduct experiments to directly compare federated with purely local training. Intuitively, federated learning should perform better since information is shared between clients, but larger PTFs may approach the performance



(a) Without task-sharing (b) Share modules in initial layers (c) Share modules in all layers

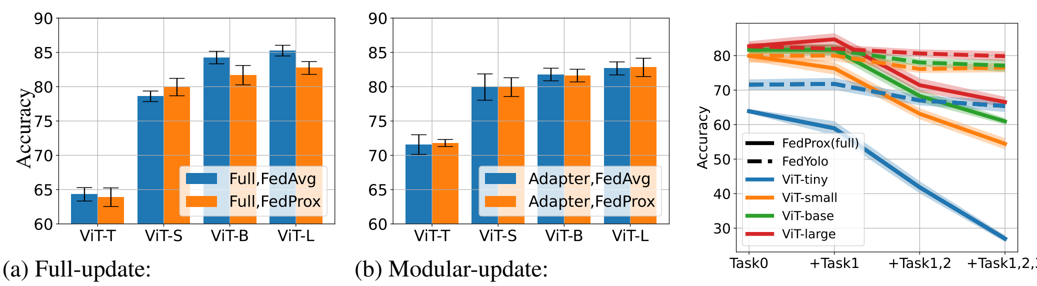
Figure 10: The figure illustrates the progression of `FedYolo` with increasing numbers of task-sharing modules from left to right. (a) Represents the vanilla `FedYolo` with independent modules for each task.(b) The modules in the initial half of the layers are shared among tasks.(c) All the modules are shared among tasks. In (b,c), the results are averages derived from three separate runs.



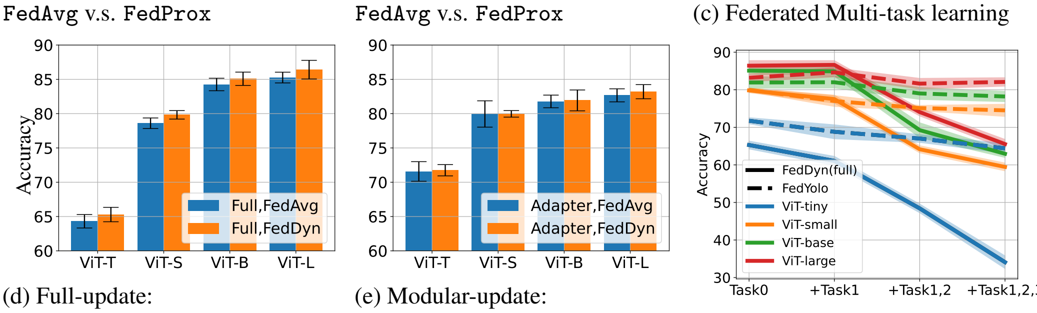
(a) `FedAvg`: Full-update vs. Modular-update (b) Full-update: `FedAvg` vs. `Local-only` (c) Modular-update: `FedAvg` vs. `Local-only`

Figure 11: Accuracy as a function of the number of training samples per class (CIFAR-100, all clients with 100 classes). (a): Larger PTFs improve accuracy for both modular-update (dashed) and full-update (solid) training strategies in the federated setting. (b,c): Comparing a federated setting (`FedAvg`, dashed) with a purely local setting (`Local-only` learning, solid), larger PTFs reduce the performance gap, especially with the modular-update training strategy.

of federated learning. In Figs. 11b,11c, we compare the performance of `Local-only` learning and `FedAvg` for the full-update and modular-update training strategies. The results show that `Local-only` learning becomes increasingly competitive with `FedAvg` as the model scale increases (i.e., the gap between the solid and dashed lines is smaller for larger PTFs). For instance, in the case of the $modular - update$ training strategy with 16 samples per client in Fig. 11c, the accuracy gap between the largest PTF (ViT-L, red line) for the `Local-only` learning and `FedAvg` strategies is 8.10, while for the smallest PTF (ViT-T, blue line), the gap is 15.44. Moreover, employing modules can help achieve better performance and narrow the gap between fully local and federated training (the gap between `Local-only` learning and `FedAvg` is smaller in Fig. 11c than Fig. 11b). In other words, if clients wish to completely avoid federated learning, large PTFs with modular updates that are trained on-device only can achieve reasonable performance.



(a) Full-update: `FedAvg` v.s. `FedProx` (b) Modular-update: `FedAvg` v.s. `FedProx` (c) Federated Multi-task learning



(d) Full-update: `FedAvg` v.s. `FedDyn` (e) Modular-update: `FedAvg` v.s. `FedDyn` (f) Federated Multi-task learning

Figure 12: Experiment results using state-of-the-art optimization-based federated learning method `FedProx` and `FedDyn`, instead of `FedAvg`. (a,b) display the model performance with a heterogeneous distribution for CIFAR-100, following the same setting as in Figure 2(left). We compare the performance between `FedProx` (orange) and `FedAvg` (blue) for different update strategies (full-update and modular-update). The results show that `FedProx` does not demonstrate notable improvement. In(c), we present the model performance in the context of federated multi-task learning. Our proposed method, `FedYolo`, consistently outperforms `FedProx` with full-update. Similar results were obtained for `FedDyn` in figures (d,e,f).

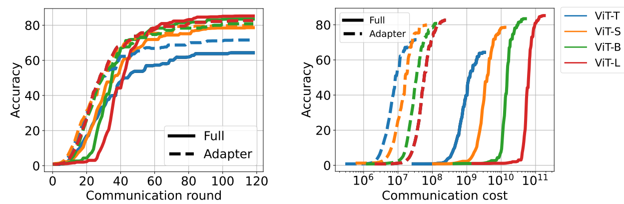
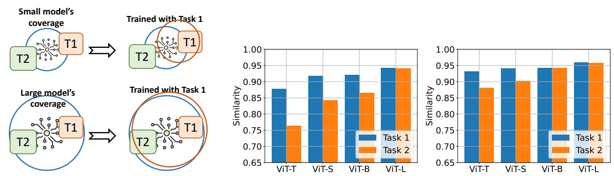


Figure 13: Communication cost is greatly decreased with modular-update compared to full-update.



(a) Representation shift (b) 100 epochs (c) 5 epochs

Figure 14: **(a)** We fine-tune a model (pre-trained on ImageNet-21k) on a specific task, Task 1. An additional Task 2, not involved in training, is also shown. We depict the representation shift (blue→red bubbles) as the model is tuned. Larger transformer has better coverage and robust to shift. **(b&c)** In our experiments, we fine-tuned few-shot CIFAR-100 dataset (400 samples), as Task 1, using modular-update. We assessed the cosine similarities of feature embedding before and after fine-tuning on CIFAR-100 (Task 1) and CIFAR-10 (Task 2) test sets. These demonstrate that feature similarity increases with larger scale and the similarity on small models declines more sharply with additional training epochs.

C.2 Reducing Federated Communication Cost

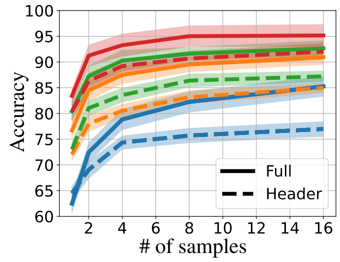
We aim to reduce communication cost while preserving accuracy, an objective intuitively achieved through modules that require fewer parameters for training than updating the entire model. Table 2 shows the number of transmitted parameters P, which is much smaller for modular-update compared to full-update across all ViT models. A consistent learning rate is maintained for a fair comparison. The experiments are conducted with mild heterogeneous CIFAR100.

Modularity Decreases Communication Rounds: We compare the number of FL communication rounds required by modular-update and full-update, plotted in Fig 13 (left). The modular-update approach (dashed lines) outperforms full-update (solid line) during initial training stages and achieves target accuracy in fewer epochs 37.25 ($\pm$1.08) on average for modular-update, versus 47.25 ($\pm$1.48) for full-update. Contrary to previous studies [15, 4], we find that modular updates typically converge faster in the federated setting.

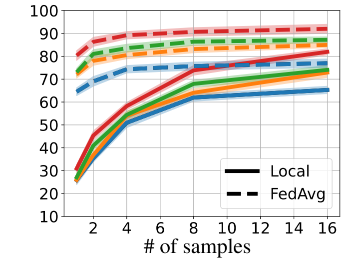
Modularity significantly reduces communication cost, by over 100x: The communication cost can be defined as $T \times M \times P$, where T is the number of communication rounds, M is the number of clients per round, and P is the number of transmitted parameters. We plot the accuracy as a function of communication cost in Fig 13 (right). modular-update significantly reduces the number of transmitted parameters compared to full-update, for all model sizes. This improvement in efficiency helps address the communication bottleneck commonly associated with federated learning.

C.3 Experiments of the Representation-based Explanation

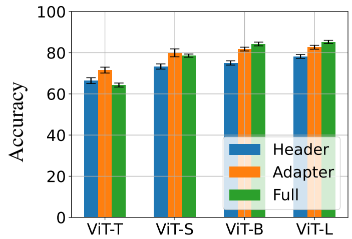
Fig. 14b&14c. We utilized cosine similarity to quantify the changes in feature embeddings throughout the fine-tuning process with different PTF scales. These figures confirm the hypothesis and show that larger models indeed incur much smaller feature changes. Also, comparing Fig. 14b&14c, features of larger model does not incur significant shift even after 100 epochs! Intuitively, this is because, the fine-tuned model is already close to the initial model in the representation space (big blue bubble mostly subsumes T1), Thus, the model fully converges in 5-10 iterations and, 100 iterations doesn't cause further shift. Finally, in additional experiments (see supplementary), we have found that, conducting the same evaluations with full-update results in consistently smaller similarities than with modular-update. This offers a potential explanation for the robustness and few-shot benefits of modular-update over full-update.



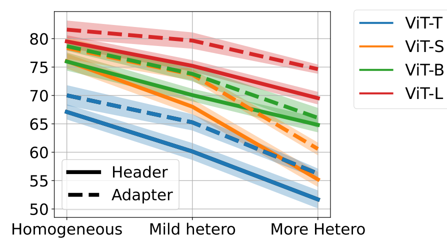
(a) Homogeneous, `FedAvg`: full-update vs. head-tuning



(b) Homogeneous, head-tuning: `FedAvg` vs. `Local-only`



(c) Heterogeneous, `FedAvg`



(d) Robustness

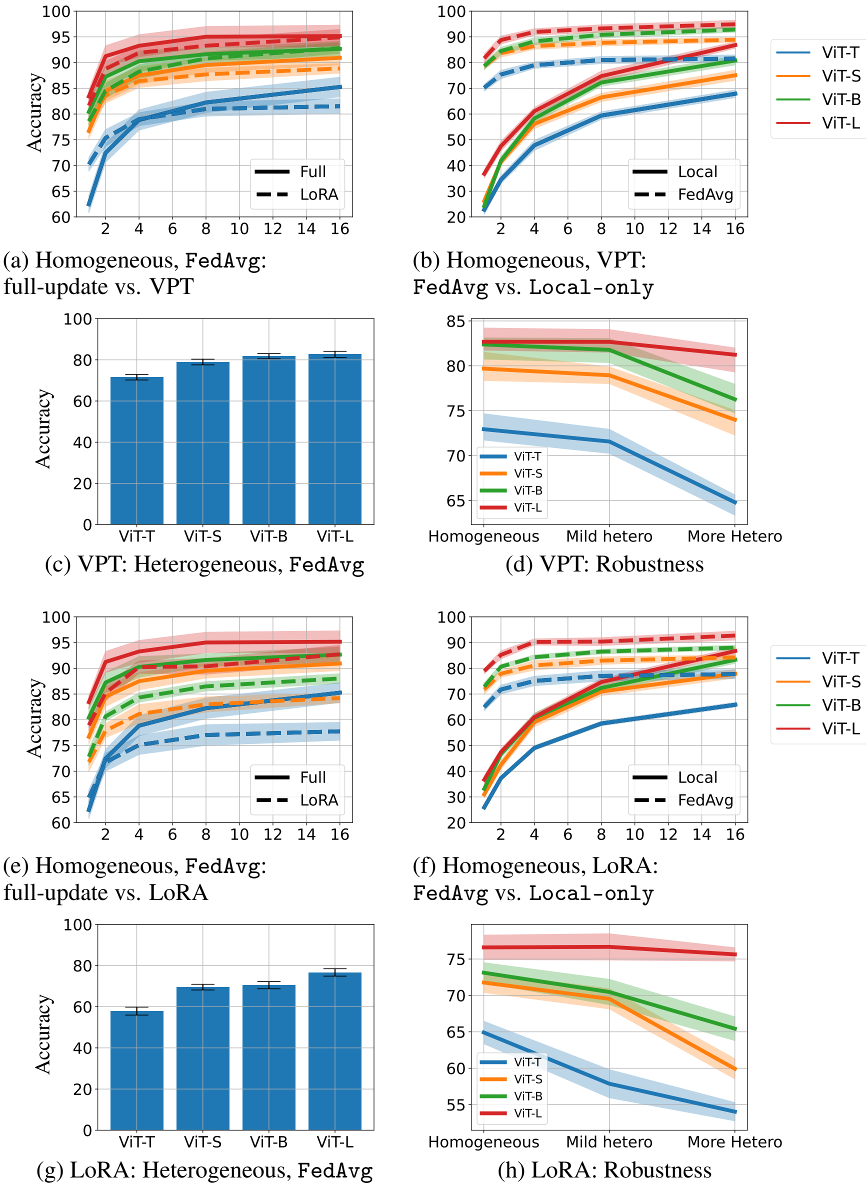
Figure 15: The results of head-tuning. (a,b) Accuracy as a function of the number of training samples per class (CIFAR-100, all clients with 100 classes). Same as the setting in Fig. 11(a) Comparing head-tuning (dashed) and full-update (solid) training strategies in the federated setting. (b) Comparing the `FedAvg` (dashed) with `Local-only` learning(solid). (c,d) Experiments are conducted with the mild heterogeneous CIFAR-100 dataset. (c) Model performance of `FedAvg`, with heterogeneous data distribution. Same as the setting in Fig. 2. modular-update consistently outperforms head-tuning in terms of performance. (d) Test accuracy under different levels of data heterogeneity. Same as the setting in Fig. 4(right). Comparing the proportion of the same curve's descent from left to right, we observe that modular-update (dashed) can achieve performance compared to head-tuning (solid).

C.4 Comparisons to Existing FL Methods

We also compare our proposed method to the state-of-the-art optimization-based federated learning method `FedProx` [21] and `FedDyn`[1]. `FedProx` uses a proximal term in the local objective function to mitigate weight divergence issues. We keep all the hyperparameters and set the penalty constant μ in the proximal term of `FedProx` to 0.1. We tune the hyperparameter μ using ViT-B and modular-update with a grid search approach and then apply the same value to all the other scales of PTFs and update strategy (full-update). The results are shown in Fig. 12. `FedProx` does not show a significant improvement in the model's performance compared to `FedAvg`. Our method `FedYolo` continues to demonstrate a substantial advantage. We conclude that `FedYolo` outperforms recent methods designed for federated learning, offering superior performance without the need for fine-tuning optimization parameters. `FedDyn` propose a dynamic regularizer for each device at each round. The results are similar. It should also be mentioned that `FedYolo` can be easily combined with those optimization-based methods.

C.5 Comparison to Head-tuning

We also evaluate the performance of the head-tuning method and compare it with modular-update in our experiments. The results can be found in Figure 15. Among all the settings, the results show that the observations from the modular-update experiments also hold true for head-tuning. Furthermore, the results consistently demonstrate that modular-update outperforms head-tuning. Figures 15b and 15a illustrate the model performance with homogeneous data, while Figures 15c and 15d compare the model performance and robustness under the heterogeneous setting. In addition to the previous observation, we find that the head-tuning performs worse than modular-update in terms of both performance and robustness.



(a) Homogeneous, `FedAvg`: full-update vs. VPT

(b) Homogeneous, VPT: `FedAvg` vs. `Local-only`

(c) VPT: Heterogeneous, `FedAvg`

(d) VPT: Robustness

(e) Homogeneous, `FedAvg`: full-update vs. LoRA

(f) Homogeneous, LoRA: `FedAvg` vs. `Local-only`

(g) LoRA: Heterogeneous, `FedAvg`

(h) LoRA: Robustness

Figure 16: The results of other modules,VPT(a-d) and LoRA(e-h). (a,b,e,f) Accuracy as a function of the number of training samples per class (CIFAR-100, all clients with 100 classes). Same as the setting in Fig. 11. (c,d) Experiments are conducted with the mild heterogeneous CIFAR-100 dataset.

C.6 Results of Other Modules

In Figure 16, we present the results of VPT and LoRA. While the type of module does influence the performance, our main findings generalize well across module types and experiments.

C.7 The Impact of Trainable Parameter Count

To verify that our empirical findings indeed arise from large-scale and modularity rather than other factors, we conduct ablation experiments. Due to the nature of module architectures like LoRA, fixed-size modules across different backbone sizes were not feasible. We use fixed dimension instead of fixed # of parameters across different scales of PTFs for a fair comparison. We explored the influence of the # of parameters in the modules using the VPT method. Among the modules, the dimensions of prompts are flexible and can be adjusted accordingly. We vary the dimensions of the VPT while keeping the total number of parameters equal to that of the ViT-L used in our experiments (299,108 parameters). In [20], the results indicate that there is a saturation point of prompt size in performance improvement. Beyond that value, further increasing the prompt size does not lead to a significant improvement in performance. Our results(shown in Fig. 17) also verify the same conclusion. This finding suggests that the advantage of larger PTFs is not attributed to the larger number of parameters.

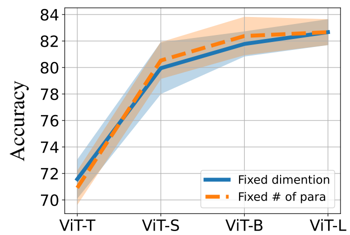
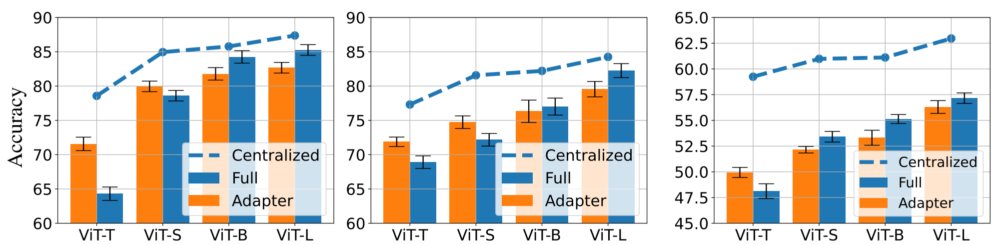


Figure 17: Our experiments demonstrate that increasing the number of parameters for smaller PTFs does not necessarily lead to improved performance. This finding suggests that the advantage of larger PTFs is not due to the larger number of parameters.

C.8 Comparison to Centralized Training

In order to assess the impact of heterogeneity on model performance, we also compare the federated accuracies to the centralized accuracies. The results are shown in Figure 18, where we observe that models with larger scale exhibit greater robustness to heterogeneity, consistent with our previous findings.



(a) CIFAR-100 (b) CelebA (c) FEMNIST

Figure 18: Comparison of different models with `FedAvg` aggregation and centralized training. The dashed line corresponds to the baseline of full-update centralized training.

C.9 The Impact of Pretraining

Previous works [30, 25] experimentally show that using pretrained models could achieve better performance compared to the models trained from scratch for federated learning settings. Our experiments align with these findings and further indicate that larger models tend to benefit more in scenarios where few-shot training is employed. The results are shown in Fig. 19. We apply `FedAvg` as the training algorithm and test on CIFAR100 with varying levels of heterogeneity.

C.10 Experiments with other datasets or training strategies

The results are shown in Fig. 20,21

D Experiment details and reproducibility

We employed a linear learning rate with linear warm-up and cosine decay scheduler for our experiments. In all federated learning methods, we set the local training epoch (E) to 1 (unless otherwise specified) and the total communication rounds to 150. We used the stochastic gradient descent (SGD)

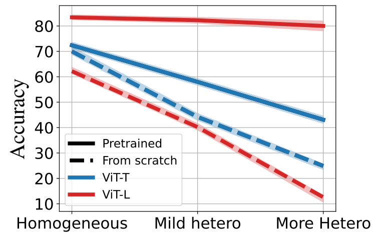
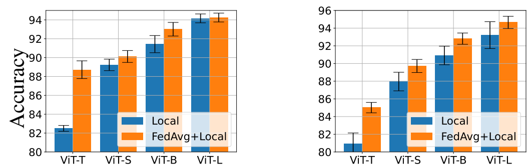


Figure 19: Our experiments confirm that employing pretrained models in federated learning leads to improved performance compared to models trained from scratch. Furthermore, our findings show that larger-scale models benefit more significantly from pretraining.



(a) CelebA, modular-update (b) CelebA,full-update

Figure 20: CelebA results for Fig. 3

optimizer with momentum of 0.9 and no weight decay. The local training batch size was set to 32, and the input image resolution was fixed at 224×224 for all methods. For CIFAR experiments, we randomly sampled 5 clients per round, while for FEMNIST and CelebA, we randomly sampled 10% of clients per round. All experiments were conducted on Tesla V100 or A100 GPU. All the experiments were run for 5 independent runs.

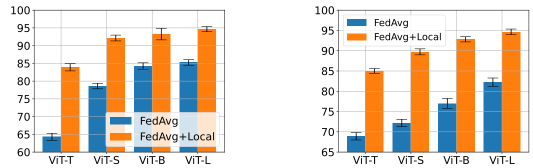
D.1 Data partition

• *CIFAR-10 and CIFAR-100:* For federated learning, we have 20 clients inspired from the settings of [25, 26]. To explore the performance under a limited sample size, we utilize a subset of the original training dataset. Experiments are conducted in both homogeneous and heterogeneous settings, where in the homogeneous setting, each client contains samples from all classes, and in the heterogeneous setting, each client contains samples from a subset of classes. We simulate three data partitions and control the non-IID level by changing the number of classes included in each client. For the CIFAR-100 dataset, the "mild heterogeneous" data partition denotes 20 classes per client, while the "more heterogeneous" data partition denotes 5 classes per client. To ensure fair comparison across data partitions and meet the challenge of limited local data, we assign 100 samples to each client, regardless of the degree of heterogeneity. The data distribution of each local test set matches that of the local train set for each client. Further details are in the supplementary. The details of data partition are provided in Fig. 22.

• *CelebA and FEMNIST:* For CelebA, we partition the dataset onto the clients based on the celebrity in each photo and test on the binary classification task of smile presence. For FEMNIST, we partition the data based on the writer of the digit/character. In accordance with [3, 25], we increase the task difficulty by dropping clients with large number of samples (specifically, 8 samples for CelebA and 120 samples for FEMNIST). For each client, we partition the data into equal 50/50 train/test sets, so the class distribution of each local test set matches that of the local train set for each client.

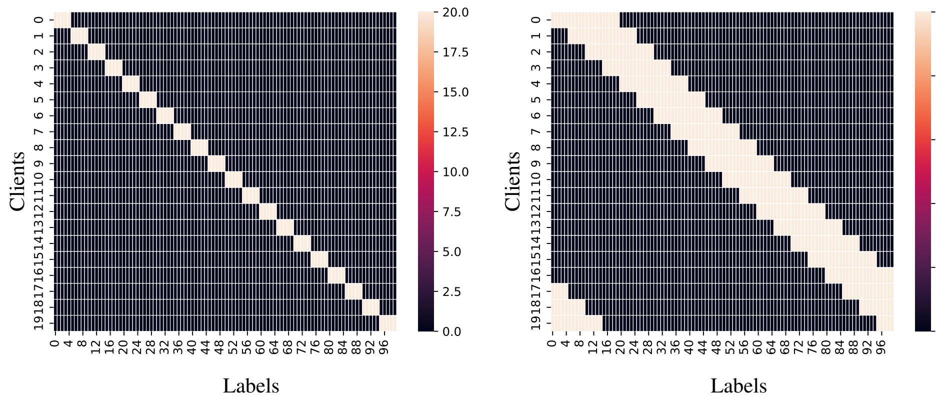
D.2 Pre-trained Transformer (PTFs):

In this study, all methods except for full-update, employed frozen PTF backbones. We utilized different scales of the Vision Transformer (ViT) architecture: ViT-large (ViT-L), ViT-base (ViT-B), ViT-small (ViT-S), and ViT-tiny (ViT-T). The models are pre-trained on ImageNet-21K from the official Google JAX implementation [6, 27, 31]. A dataset-specific header is deployed to adapt to the number of classes for each dataset. The number of trainable parameters for is shown in Table. 2. For other training strategies, the number of trainable parameters is available in the supplementary.

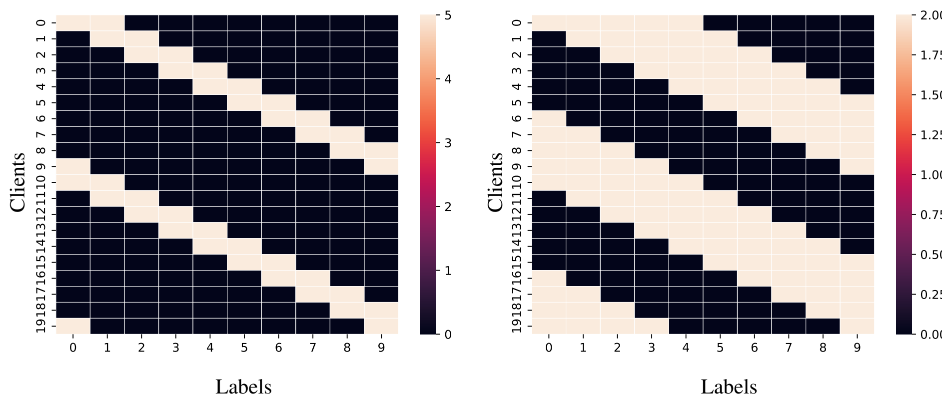


(a) CIFAR100,full-update (b) CelebA, full-update

Figure 21: Test set accuracy with (`FedAvg+Local`) and without (`FedAvg`) personalization. As the PTF scale increases, the gap between the two approaches diminishes. full-update results for Fig. 5.



(a) CIFAR100, More heterogeneous (b) CIFAR100, Mild heterogeneous



(c) CIFAR10, More heterogeneous (d) CIFAR10, Mild heterogeneous

Figure 22: Data partition for different non-IID level

D.3 Modules:

We evaluated several modules for the modular-update method, including Adapter [12], LoRA [13], and VPT [16]. Due to space limitations, we only include the results of the Adapter in the main paper, while the results of the LoRA and VPT methods are similar and relegated to the supplementary material. Therefore, in the results below, the modular-update and "Adapter" terms are used interchangeably. To ensure a fair comparison, we deploy the modules on all transformer blocks, maintaining a fixed embedding dimension of 8 across different scales. The Appendix provides further details on the size of each module plus PTF. The number of trainable parameters for each training strategy was shown in Table. 3

D.4 Personalized training:

For heterogeneous data distribution (§3.2), we also perform personalized training after the global federated training. Each client will thus have its own personalized model. During the personalized

	ViT-T	ViT-S	ViT-B	ViT-L
Full-update	5.5M	21.7M	85.9M	303.4M
Modular-update	58.6K	116.9K	233.6K	418.0K

Table 2: Number of parameters for different PTF scales.

	ViT-T	ViT-S	ViT-B	ViT-L
Full model	5,543,716	21,704,164	85,875,556	303,404,132
Adapter	58,564	116,932	233,668	417,984
LoRA	93,028	185,956	371,812	888,932
VPT	37,732	75,364	150,628	299,108
Header	19,300	38,500	76,900	102,500

Table 3: Number of parameters for different PTF scales and different tuning methods.

training, we fine-tune the average global model using local data to obtain a customized model for each client.

D.5 Evaluation metrics:

Unless otherwise stated, the evaluation of all models is based on the average local accuracy across clients. In the case of `FedAvg`, the performance of the average global model is calculated and shared among all clients. For `Local-only` learning and `FedAvg+Local`, each client has its own fine-tuned model, so we compute the average performance of the individual models. In all figures, error bars correspond to one standard deviation.

D.6 Optimizers:

We use FedAvg with SGD optimizer, momentum parameter of 0.9, and no weight decay. The local training batch size is set to 32. In appendix, we also provide experiments for FedProx [21] and FedDyn [1] which led to consistent conclusions as FedAvg (see supplementary).