Profit: Benchmarking Personalization and Robustness Trade-off in Federated Prompt Tuning

Anonymous Author(s)
Affiliation
Address
email

Abstract

In many applications of federated learning (FL), clients desire models that are personalized using their local data, yet are also robust in the sense that they retain general global knowledge. However, the presence of data heterogeneity across clients induces a fundamental trade-off between personalization (i.e., adaptation to a local distribution) and robustness (i.e., not forgetting previously learned general knowledge). It is critical to understand how to navigate this personalization vs robustness trade-off when designing federated systems, which are increasingly moving towards a paradigm of fine-tuning large foundation models. Due to limited computational and communication capabilities in most federated settings, this foundation model fine-tuning must be done using parameter-efficient fine-tuning (PEFT) approaches. While some recent work has studied federated approaches to PEFT, the personalization vs robustness trade-off of federated PEFT has been largely unexplored. In this work, we take a step towards bridging this gap by benchmarking fundamental FL algorithms – FedAvg and FedSGD plus personalization (via client local fine-tuning) – applied to one of the most ubiquitous PEFT approaches to large language models (LLMs) – prompt tuning – in a multitude of hyperparameter settings under varying levels of data heterogeneity. Our results show that federated-trained prompts can be surprisingly robust when using a small learning rate with many local epochs for personalization, especially when using an adaptive optimizer as the client optimizer during federated training. We also demonstrate that simple approaches such as adding regularization and interpolating two prompts are effective in improving the personalization vs robustness trade-off in computation-limited settings with few local updates allowed for personalization.

1 Introduction

Federated learning (FL) is a framework that enables distributed clients to collaboratively train machine learning models in a privacy-preserving manner [43, 25, 33, 65]. Unlike traditional server-side distributed training, in FL, each client (e.g., a mobile device)'s local data may follow a distinct distribution. This data heterogeneity motivates the development of personalized FL: the goal is to learn client-specific models that work well for each client's own data. Among all the personalized FL approaches [e.g., 52, 57, 64, 6], one of the simplest methods is fine-tuning a global model on each client's local data to produce a personalized model [66, 24]. Despite its simplicity, fine-tuning a FedAvg (Federated Averaging [45, 43])-trained global model has connections to meta learning [24, 5] and representation learning [12], and has been shown to work well over on-device data [58, 48].

Most of the existing FL personalization benchmarks (e.g., [64, 6, 41]) focus on training small-sized models (e.g., in the order of 10M parameters) from scratch. In this paper, we consider prompt tuning a pre-trained large language model (LLM) (specifically, an 8B parame-

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

ter version of the PaLM model [10]) in the federated setting. As shown in Figure 1, similar to the setup considered in [70], during FedAvg training, the PaLM-8B model is kept frozen, and only the soft prompt part is tuned and communicated between the server and clients; and during the personalization phase, each client will fine-tune the soft prompt locally to create a personalized soft prompt. Prompt tuning [31] is one of the standard parameter-efficient fine-tuning (PEFT) algorithms [14, 36] proposed for LLMs. Considering the potential communication and memory limitations in the FL settings, PEFT is more suitable than full-model fine-tuning; besides, PEFT is shown to be capable of matching full-model fine-tuning in many scenarios [31, 21]. To create a federated dataset, similar to [67], we partition a large-scale instruction tuning dataset based on the task types. We create datasets with three different heterogeneity levels (see Figure 2 for an overview of our setup).

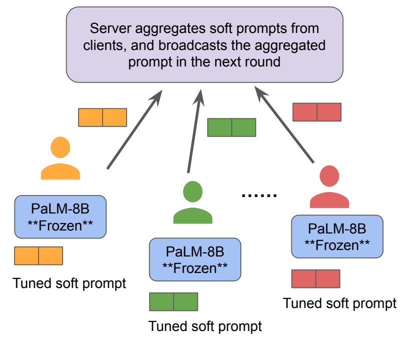

Figure 1: In each training round, only the soft prompts are updated and communicated between server and clients.

Our contributions are summarized below:

- We run comprehensive experiments to study the trade-off between personalization (adaptation to the clients' local distributions) and robustness (not forgetting the previously learned knowledge obtained during the FL training) over different FL training algorithms (variants of FedAvg and FedSGD) and different data heterogeneity levels (high/medium/low). To our knowledge, we are the first to study this trade-off in the setting of FL personalization and LLM prompt tuning.
- We observe that for federated prompt tuning, it is important to use adaptive optimizer (e.g., Adam [27]) as the client optimizer[1] in FedAvg (even though the server optimizer already uses adaptive optimizer). This is unlike previous proposed adaptive FedAvg algorithm [45] (which uses adaptive optimizer at the server, and vanilla SGD at the clients). Our hypothesis is that the loss surface is very flat due to the large scale of the learned soft prompt, so using adaptive optimizer at the clients are crucial in making enough progress during training (see Section 4 Observation 3a).
- We observe that during the personalization stage (i.e., during the local prompt fine-tuning stage), smaller learning rate achieves better personalization vs robustness trade-off, but it has to run many steps to reach the best personalization performance. We also find that simple methods such as adding regularization and/or model averaging are effective to achieve the best of both worlds: better personalization vs robustness trade-off in fewer local tuning steps (see Figure 5).

2 Related Works

Federated PEFT of pre-trained LLMs. A number of works have begun to explore PEFT in the federated settings. Some have studied federated prompt tuning on vision tasks, without evaluating personalization [69, 8, 18]. Other works have benchmarked federated PEFT on language tasks, but again did not consider personalization [67, 71, 4, 3]. To our knowledge, all studies of federated PEFT that consider personalization focus on the vision modality [17, 32, 38, 50, 70]. Outside of PEFT, [20, 53, 61] studied federated full-model fine-tuning of BERT models, which are at least an order of magnitude smaller than modern LLMs. Multiple works have noticed that initializing full-model federated training from a pre-trained model can mitigate the effects of data heterogeneity [44, 61, 7]. Like our work, [44] also noticed the importance of using adaptive optimizers when running federated fine-tuning, but they only considered full-model fine-tuning starting from small models. Other works have analyzed the effect of differential privacy on federated training of language models via initialization with [35] or by distillation from a pre-trained LLM [55] .

Personalization in FL. A long line of work within federated learning has developed techniques for personalizing models to each client [13, 19, 51, 15, 34, 39, 49, 11, 47, 40]. We defer readers to the recent FL personalization benchmarks [64, 6, 41] and the references therein for a more detailed discussion of the related work. In this paper, we focus on one of the simplest personalization

[1]Note that the resulting algorithm is still a *stateless* algorithm. A stateless algorithm means that the client does not maintain states locally and reuse them in the next participating round [25, 57, 64]. In our setting, it means that clients do not store Adam optimizer state (estimates of moments). Stateful algorithms (e.g., SCAFFOLD [26]) can perform poorly with low clients participating rate (see Section 5.1 of [45]).

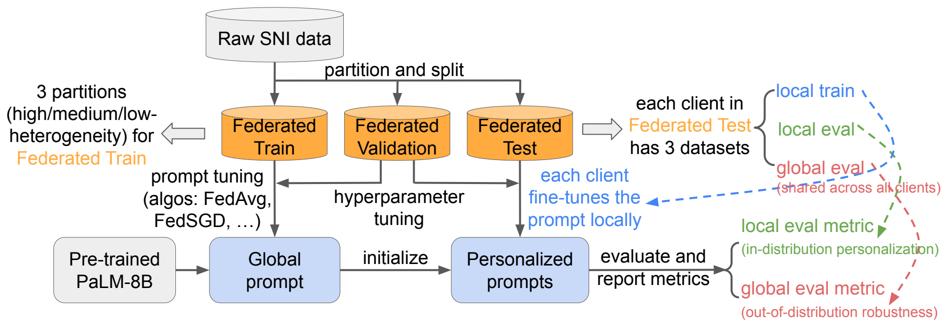

Figure 2: Overview of our experimental setup. We partition and split the raw SNI dataset into three federated datasets: train (used for training a global prompt), validation (used for hyperparameter tuning), and test (used for learning and evaluating the personalized prompts). We experiment with three versions (high/medium/low heterogeneity) of training data. In the test data, each client has three local datasets: a local train set (used for locally fine-tuning the global prompt to produce the personalized prompt) and local and global eval sets (used for evaluating the personalized prompt over the local and global distributions, respectively). The global eval set is shared across all clients, and is formed by sampling from all test clients' local eval sets. See Section 3 for more details.

approaches: each client fine-tunes a model locally to get the personalized model [66, 24, 12, 9, 5]. In particular, we are interested in studying the personalization and robustness trade-off. To our knowledge, we are the first to study this trade-off in the setting of federated prompt tuning for LLMs.

Robustness to catastrophic forgetting during fine-tuning. Robustness can have different definitions, e.g., robustness to attacks [34, 59] and outliers [30]. In this paper, we focus on a special type, that is, robustness to forgetting about the global knowledge learned by FedAvg when each client fine-tunes the global prompt locally to get a personalized prompt. This is connected to the robustness to distribution shift or out-of-distribution data in the literature, see, e.g., [1, 62, 63, 22, 54, 29, 23], where the main difference is that in our experiments, the in-distribution and out-of-distribution have a special connection unique to the FL setting: a client's local distribution vs all clients' joint distribution. Catastrophic forgetting [42] has been studied for decades. Many proposed methods (e.g., [46, 28]) may not directly fit the FL setting due to privacy or computation constraint. [48] considers a production FL scenario, and proposes to let each client to decide whether to accept the personalized model based on validation data metric. This is orthogonal to the robust fine-tuning methods we experiment with in Figure 5, where we tried two simple robust fine-tuning methods (regularization and model averaging [62, 63, 22]) that do not modify model architecture. We leave the investigation of more complicated robust fine-tuning methods (e.g., [54, 23]) to future work.

3 Experimental Setup

In this section we detail the framework we use to empirically evaluate federated-trained prompts.

Datasets. We construct three federated datasets from Super-NaturalInstructions (SNI) [60]. SNI is a collection of 1761 diverse NLP tasks belonging to one of 76 *task types*. Task types include both text classification and generation types, with Translation, Question Answering, and Question Generation being the most popular. Tasks have on average ∼3000 (query, target) pairs, called instances.

We partition the instances into clients by first splitting them into training, validation, and test sets according to task type. We randomly select 7 task types each for testing and validation[2]. Then, we partition the test and validation data into clients by ordering the instances in each task type by task, then breaking these lists into evenly-sized chunks of adjacent instances and designating each chunk to a client. As a result, each client's instances belong to a single task type, and typically a single task. Next, we construct three distinct partitions of the training data. First, we construct a *high heterogeneity* partition in exactly the same manner as we partition the validation and test data. We do the same for a *medium heterogeneity* partition, except that we shuffle the instances within each task type before dividing them into client chunks, so that each client may have instances from many tasks

[2]The test task types are Irony Detection, Text Completion, Explanation, Overlap Extraction, Question Generation, Dialogue Act Recognition, and Gender Classification.

of the same type. Lastly, we construct a *low heterogeneity* partition by shuffling the entire dataset before dividing it into client chunks, thus each client has instances from many tasks of many types. All of each training clients' instances are used in federated training, and the same validation and test sets are used for all three partitions. We call these three partitions High Heterogeneity Federated SNI (**HHF-SNI**), Medium HF-SNI (**MHF-SNI**), and Low HF-SNI (**LHF-SNI**), respectively, and provide dataset statistics that verify heterogeneity levels in Table 1 and Figure 6 in Appendix C.

Model and metric. We use the 8 billion-parameter version of the original PaLM [10], which was trained on 780 billion tokens from sources including social media and Wikipedia[3]. Following [60], we use ROUGE-L [37] to measure similarity between predicted and target sequences, with scores in [0, 1] and larger scores indicating greater similarity.

Table 1: Dataset statistics. Entries show the mean total instances and unique tasks and task types found in each client's dataset (rounded to the nearest integer) $\pm$ standard deviation across training clients. All partitions have **3520 training clients** and all federated experiments sample **32 training clients/round**. There are 326 test and validation clients each, and each has approximately 1200 instances.

Dataset	Instances	Tasks	Task types
HHF-SNI	1201 ± 17.6	1 ± 0.8	1 ± 0
MHF-SNI	1201 ± 17.6	118 ± 111.2	1 ± 0
LHF-SNI	1201 ± 0.4	640 ± 10.8	50 ± 1.8

Experimental procedure. We execute a two-stage experimental procedure. In Stage 1, we run federated learning on the training clients to learn global prompt parameters (see Appendix A for more details on prompt tuning). In Stage 2, we evaluate the quality of these global parameters by using them to initialize local training (personalization) on each test client. In particular, each test client independently trains a soft prompt on their training set starting from the federated-trained global prompt. As this local training progresses we record the prompt's scores on the corresponding client's test data and on a global test dataset compiled across all of the test clients' test datasets. The local scores serve as the personalization metric, while the global scores serve as the robustness metric. We hyperparameter tune in Stage 1 by evaluating the global prompt on a global validation dataset collected from all the validation clients, and in Stage 2 by running personalization on the validation clients. Figure 2 depicts this procedure in detail.

Baselines and hyperparameters. We study a generalized version of FedAvg proposed in [45] that allows for adaptive server and client optimizers[1]. As in [45], we find that using an adaptive server optimizer, in our case Adam, improves over SGD, so all our experiments use Adam on the server side. For the client optimizer[1], we experiment with both Adam and SGD, referring to these versions of FedAvg as **FedAvg(Adam)** and **FedAvg(SGD)**, respectively. Both algorithms make 16 local updates with batch size 32 on 32 sampled clients per round for 300 rounds, and the Adam optimizer is re-initialized from scratch at the start of each selected client's local training round. We also consider **FedSGD**, in which 32 clients per round send the gradient of the global prompt estimated on 32 instances directly back to the server, and the server updates the global model using Adam. We execute FedSGD for 4800 rounds so that FedSGD processes the same total number of instances as the FedAvg methods. In Appendix C, we explore a version of FedSGD that multiplies the batch size (rather than the number of communication rounds) by 16 in order to see the same number of instances as FedAvg, noting that this gave significantly worse results. We also run **Centralized** training with Adam and batch size 1024 (same effective batch size as FedSGD) for 4800 rounds.

All algorithms optimize prompts of length 10 (tuned in $\{5, 10, 20\}$) with embedding dimension 4096. We tune learning rates, the Adam epsilon parameter, and the weight decay parameter during federated training. For personalization, we run Adam and tune its learning rate based on the number of epochs available. We evaluate on 32 test clients, each with training and test sets of 256 and 128 instances, respectively, and a global test set of 2048 instances. Additional details are provided in Appendix C.

4 Results

Next, we share personalization (i.e., the local score obtained by evaluating a client's personalized model on this client's local data) vs robustness (i.e., the global score obtained by evaluating the same personalized model over the global test set) curves during personalization. Each point in each plot

[3]We choose this model to minimize data leakage, since it was released prior to the release of SNI. Nevertheless, there could still be overlap between its training data and the sources used by SNI.

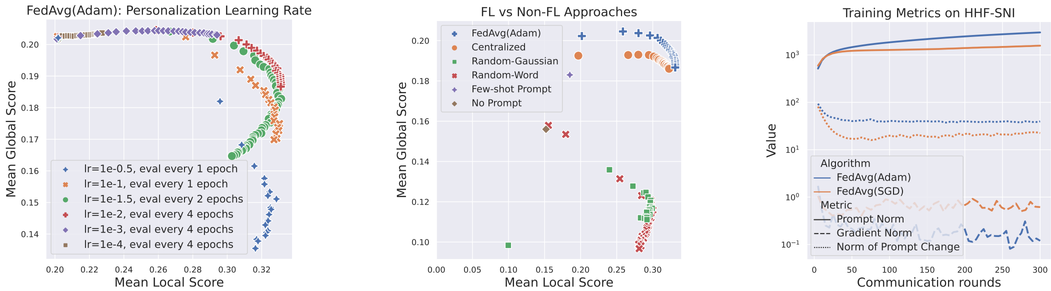

Figure 3: (Left) Global and local scores during personalization with varying learning rates from a prompt trained on HHF-SNI by FedAvg(Adam). All runs besides those with the largest two learning rates are run for 100 epochs, and otherwise 20 epochs. **(Center)** Global and local scores during 100 epochs (high computation) of personalization starting from FedAvg(Adam) and Centralized-pre-trained prompts and random initializations (with evaluations every 4 epochs), plus global and local scores with no prompt and few-shot (engineered) prompts. **(Right)** Global prompt norm, average gradient norm across clients, and norm of prompt change on consecutive rounds during FedAvg(Adam) and FedAvg(SGD) training. All norms are Frobenius.

is the mean (local score, global score) across clients during a personalization epoch, averaged over two-end-to-end trials with distinct random seeds[4]. These results admit a number of observations.

Observation 1: Choice of personalization learning rate induces computation vs robustness trade-off. Figure 3(Left) plots global and local scores during personaliztion with varying learning rates starting from a prompt pre-trained on HHF-SNI with FedAvg(Adam). These results show that the personalization vs robustness trade-off is heavily dependent on the personalization learning rate. In particular, higher global scores can be maintained by personalizing with smaller learning rates, but at the cost of requiring more epochs to reach the maximal local scores. Specifically, with learning rate $10^{-0.5}$, the average local score reaches 0.32 within 10 epochs and the average global score drops to 0.15, and with learning rate 10^{-2}, 64 epochs are required to reach average local score 0.32, but the average global score does not drop below 0.19. In effect, this induces a computation vs robustness trade-off: more robustness necessitates more computation.

This motivates us to consider two distinct regimes for personalization: (1) **High Computation**, in which each client executes 100 epochs of personalization, and (2) **Low Computation**, in which each client executes 10 epochs of personalization, with learning rates tuned to achieve the best local score (0.32) with minimal drop in global score for each regime. We use regime (1) to compare different pre-training algorithms, as this allows the best performance for each algorithm (Observations 2 and 3). Then, we conclude by showing the more severe forgetting in regime (2) can be mitigated by incorporating a number of heuristics (Observation 4).

Observation 2: Benefit of FL pre-training. Figure 3(Center) considers the High Computation regime and shows global vs local score curves for prompts pre-trained with FedAvg(Adam) and centralized training, along with prompts initialized by sampling from a Gaussian distribution ("Random Gaussian") and by sampling 10 token embeddings from the PaLM token embedding matrix ("Random Word") [16]. FedAvg(Adam) yields the best personalization vs robustness trade-off, especially compared to the random initializations. Surprisingly, FedAvg(Adam) outperforms centralized training, although centralized training achieves smaller training loss (see Appendix C), as expected due to possible objective inconsistency for FedAvg [56]. FedAvg(Adam) also outperforms both No Prompt and Few-shot Prompts, which are constructed using instructional examples according to the best procedure reported in [60]; please see Appendix C for details.

Observation 3a: Importance of adaptive client optimizer[1]. Figure 4 compares prompts trained with FedAvg(Adam), FedAvg(SGD), and FedSGD during personalization in the High Computation regime. FedAvg(Adam) outperforms FedAvg(SGD) on all three training partitions, highlighting the benefit of using an adaptive client optimizer[5]. It is well-known that adaptive optimization enhances full-model transformer training [68], but to our knowledge this has not yet been observed for prompt

[4]Our observations are consistent across random seeds; see results for individual seeds in Appendix C.

[5]Often, the client optimizer in FL is SGD, motivated by the added memory cost of Adam [45]. However, this cost is linear in the number of trainable parameters, so it is small for prompt tuning.

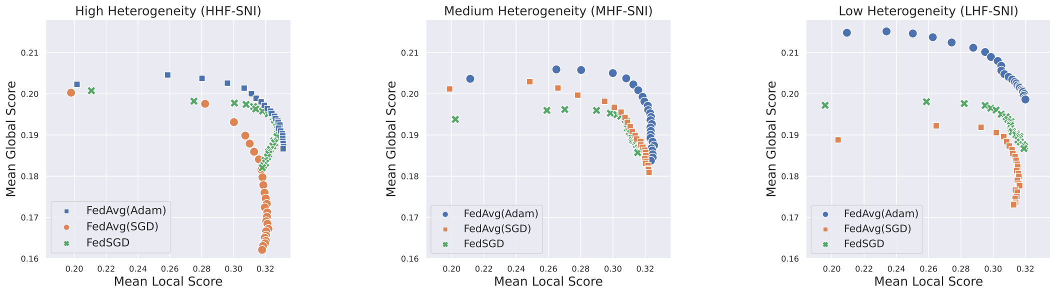

Figure 4: High Computation regime: scores evaluated every 4 epochs during **100 epochs** of personalization starting from prompts pre-trained by FedAvg(Adam), FedAvg(SGD) and FedSGD on **(Left)** HHF-SNI, **(Center)** MHF-SNI, and **(Right)** LHF-SNI.

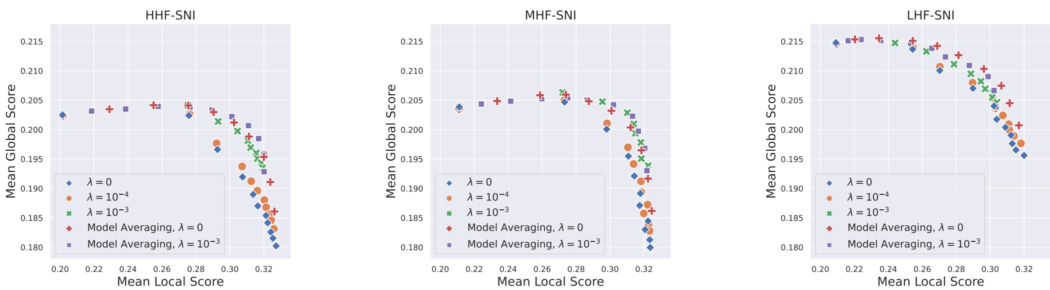

Figure 5: Low Computation regime: scores evaluated every epoch during **10 epochs** of personalization with robust-l2 regularization with parameter λ, and possibly model averaging, starting from prompts trained by FedAvg(Adam) on **(Left)** HHF-SNI, **(Center)** MHF-SNI, and **(Right)** LHF-SNI.

tuning. Based on Figure 3, we conjecture that Adam's benefit stems from prompt tuning's flat loss landscape relative to prompt scale. For both FedAvg(Adam) and FedAvg(SGD), gradient norms are three orders of magnitude smaller than prompt norms throughout training. This means that the SGD updates are relatively insignificant, unlike the Adam updates that have normalized gradient and a momentum term that scales with the prompt norm. Thus, FedAvg(SGD) has smaller prompt changes than FedAvg(Adam), despite having a client learning rate 100x larger (see Table 3).

Observation 3b: Importance of multiple local updates. Figure 4 also shows that FedAvg(Adam) outperforms FedSGD, especially with lower training data heterogeneity. Multiple recent works have noticed the superiority of FedAvg-trained models as initializations for personalization compared to FedSGD-trained models [5, 12, 24], but these works did not consider the robustness to forgetting after personalization (nor prompt tuning). In contrast, here we observe that the improvement due to FedAvg is mostly due to higher *global scores*. Since we use Adam as the server optimizer for FedSGD, the improvement of FedAvg(Adam) cannot be due to its updates being adaptive, but must be due to making multiple of them between communication.

Observation 4: Personalization-robustness trade-off can be improved by personalization heuristics. Figure 5 considers the Low Computation regime, in which each client only executes 10 personalization epochs. Here, we evaluate two heuristics to improve the personalization vs robustness trade-off: (1) l2 regularization and (2) model averaging [62, 63, 22]. For (1), we add l2 regularization with parameter λ to the loss that penalizes the distance of the personalized prompt from the global prompt. For (2), we first run full personalization, then compute final client-specific prompts by interpolating the global and personalized prompts, with increasing weight on the personalized prompt moving from left to right in the plots. Figure 5 shows that both of these techniques, as well as their combination, improve the personalization-robustness trade-off for FedAvg(Adam)-trained prompts.

Conclusion. Our benchmarking experiments evince the effectiveness of FL for prompt pre-training. We also provide methods to improve the personalization vs robustness trade-off for federated-trained prompts. Nevertheless, we only explore simple FL algorithms, without privacy guarantees, on a single model (PaLM-8b); investigation of federated prompt tuning's performance along each of these axes remains important future work.

References

[1] Anders Andreassen, Yasaman Bahri, Behnam Neyshabur, and Rebecca Roelofs. The evolution of out-of-distribution robustness throughout fine-tuning. *arXiv preprint arXiv:2106.15831*, 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] Dongqi Cai, Yaozong Wu, Shangguang Wang, Felix Xiaozhu Lin, and Mengwei Xu. Fedadapter: Efficient federated learning for modern nlp. *arXiv preprint arXiv:2205.10162*, 2022.

[4] Dongqi Cai, Yaozong Wu, Haitao Yuan, Shangguang Wang, Felix Xiaozhu Lin, and Mengwei Xu. Aug-fedprompt: Practical few-shot federated nlp with data-augmented prompts. *arXiv preprint arXiv:2212.00192*, 2022.

[5] Zachary Charles, Nicole Mitchell, Krishna Pillutla, Michael Reneer, and Zachary Garrett. Towards federated foundation models: Scalable dataset pipelines for group-structured learning. *arXiv preprint arXiv:2307.09619*, 2023.

[6] Daoyuan Chen, Dawei Gao, Weirui Kuang, Yaliang Li, and Bolin Ding. pfl-bench: A comprehensive benchmark for personalized federated learning. In *NeurIPS Datasets and Benchmarks track*, 2022.

[7] Hong-You Chen, Cheng-Hao Tu, Ziwei Li, Han-Wei Shen, and Wei-Lun Chao. On pre-training for federated learning. *arXiv preprint arXiv:2206.11488*, 2022.

[8] Jinyu Chen, Wenchao Xu, Song Guo, Junxiao Wang, Jie Zhang, and Haozhao Wang. Fedtune: A deep dive into efficient federated fine-tuning with pre-trained transformers. *arXiv preprint arXiv:2211.08025*, 2022.

[9] Gary Cheng, Karan Chadha, and John Duchi. Fine-tuning is fine in federated learning. *arXiv preprint arXiv:2108.07313*, 3, 2021.

[10] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

[11] Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. Exploiting shared representations for personalized federated learning. In *International conference on machine learning*, pages 2089–2099. PMLR, 2021.

[12] Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. Fedavg with fine tuning: Local updates lead to representation learning. *Advances in Neural Information Processing Systems*, 35:10572–10586, 2022.

[13] Yuyang Deng, Mohammad Mahdi Kamani, and Mehrdad Mahdavi. Adaptive personalized federated learning. *arXiv preprint arXiv:2003.13461*, 2020.

[14] 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.

[15] 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.

[16] Yuxian Gu, Xu Han, Zhiyuan Liu, and Minlie Huang. Ppt: Pre-trained prompt tuning for few-shot learning. *arXiv preprint arXiv:2109.04332*, 2021.

[17] Tao Guo, Song Guo, and Junxiao Wang. pfedprompt: Learning personalized prompt for vision-language models in federated learning. In *Proceedings of the ACM Web Conference 2023*, pages 1364–1374, 2023.

[18] Tao Guo, Song Guo, Junxiao Wang, Xueyang Tang, and Wenchao Xu. Promptfl: Let federated participants cooperatively learn prompts instead of models-federated learning in age of foundation model. *IEEE Transactions on Mobile Computing*, 2023.

[19] Filip Hanzely and Peter Richtárik. Federated learning of a mixture of global and local models. *arXiv preprint arXiv:2002.05516*, 2020.

[20] Agrin Hilmkil, Sebastian Callh, Matteo Barbieri, Leon René Sütfeld, Edvin Listo Zec, and Olof Mogren. Scaling federated learning for fine-tuning of large language models. In *International Conference on Applications of Natural Language to Information Systems*, pages 15–23. Springer, 2021.

[21] 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. *arXiv preprint arXiv:2106.09685*, 2021.

[22] Gabriel Ilharco, Mitchell Wortsman, Samir Yitzhak Gadre, Shuran Song, Hannaneh Hajishirzi, Simon Kornblith, Ali Farhadi, and Ludwig Schmidt. Patching open-vocabulary models by interpolating weights. *Advances in Neural Information Processing Systems*, 35:29262–29277, 2022.

[23] Liangze Jiang and Tao Lin. Test-time robust personalization for federated learning. In *International Conference on Learning Representations*, 2023.

[24] Yihan Jiang, Jakub Konečný, Keith Rush, and Sreeram Kannan. Improving federated learning personalization via model agnostic meta learning. *arXiv preprint arXiv:1909.12488*, 2019.

[25] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends in Machine Learning*, 14(1–2):1–210, 2021.

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

[27] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[28] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[29] Ananya Kumar, Aditi Raghunathan, Robbie Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *International Conference on Learning Representations*, 2022.

[30] Achintya Kundu, Pengqian Yu, Laura Wynter, and Shiau Hong Lim. Robustness and personalization in federated learning: A unified approach via regularization. In *2022 IEEE International Conference on Edge Computing and Communications (EDGE)*, pages 1–11, 2022. doi: 10.1109/EDGE55608.2022.00014.

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

[32] Guanghao Li, Wansen Wu, Yan Sun, Li Shen, Baoyuan Wu, and Dacheng Tao. Visual prompt based personalized federated learning. *arXiv preprint arXiv:2303.08678*, 2023.

[33] 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.

[34] Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. Ditto: Fair and robust federated learning through personalization. In *International Conference on Machine Learning*, pages 6357–6368. PMLR, 2021.

[35] Xuechen Li, Florian Tramer, Percy Liang, and Tatsunori Hashimoto. Large language models can be strong differentially private learners. *arXiv preprint arXiv:2110.05679*, 2021.

[36] Vladislav Lialin, Vijeta Deshpande, and Anna Rumshisky. Scaling down to scale up: A guide to parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.15647*, 2023.

[37] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013.

[38] Wang Lu, Xixu Hu, Jindong Wang, and Xing Xie. Fedclip: Fast generalization and personalization for clip in federated learning. *arXiv preprint arXiv:2302.13485*, 2023.

[39] Yishay Mansour, Mehryar Mohri, Jae Ro, and Ananda Theertha Suresh. Three approaches for personalization with applications to federated learning. *arXiv preprint arXiv:2002.10619*, 2020.

[40] Othmane Marfoq, Giovanni Neglia, Richard Vidal, and Laetitia Kameni. Personalized federated learning through local memorization. In *International Conference on Machine Learning*, pages 15070–15092. PMLR, 2022.

[41] Koji Matsuda, Yuya Sasaki, Chuan Xiao, and Makoto Onizuka. An empirical study of personalized federated learning. *arXiv preprint arXiv:2206.13190*, 2022.

[42] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

[43] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[44] John Nguyen, Kshitiz Malik, Maziar Sanjabi, and Michael Rabbat. Where to begin? exploring the impact of pre-training and initialization in federated learning. *arXiv preprint arXiv:2206.15387*, 2022.

[45] Sashank J. Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and Hugh Brendan McMahan. Adaptive federated optimization. In *International Conference on Learning Representations*, 2021.

[46] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[47] Aviv Shamsian, Aviv Navon, Ethan Fetaya, and Gal Chechik. Personalized federated learning using hypernetworks. In *International Conference on Machine Learning*, pages 9489–9502. PMLR, 2021.

[48] Khe Chai Sim, Angad Chandorkar, Fan Gao, Mason Chua, Tsendsuren Munkhdalai, and Françoise Beaufays. Robust continuous on-device personalization for automatic speech recognition. In *Interspeech*, pages 1284–1288, 2021.

[49] Karan Singhal, Hakim Sidahmed, Zachary Garrett, Shanshan Wu, John Rush, and Sushant Prakash. Federated reconstruction: Partially local federated learning. *Advances in Neural Information Processing Systems*, 34:11220–11232, 2021.

[50] Shangchao Su, Mingzhao Yang, Bin Li, and Xiangyang Xue. Cross-domain federated adaptive prompt tuning for clip. *arXiv preprint arXiv:2211.07864*, 2022.

[51] Canh T Dinh, Nguyen Tran, and Josh Nguyen. Personalized federated learning with moreau envelopes. *Advances in Neural Information Processing Systems*, 33:21394–21405, 2020.

[52] Alysa Ziying Tan, Han Yu, Lizhen Cui, and Qiang Yang. Towards personalized federated learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[53] Yuanyishu Tian, Yao Wan, Lingjuan Lyu, Dezhong Yao, Hai Jin, and Lichao Sun. Fedbert: When federated learning meets pre-training. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 13(4):1–26, 2022.

[54] Dustin Tran, Jeremiah Liu, Michael W Dusenberry, Du Phan, Mark Collier, Jie Ren, Kehang Han, Zi Wang, Zelda Mariet, Huiyi Hu, et al. Plex: Towards reliability using pretrained large model extensions. *arXiv preprint arXiv:2207.07411*, 2022.

[55] Boxin Wang, Yibo Jacky Zhang, Yuan Cao, Bo Li, H Brendan McMahan, Sewoong Oh, Zheng Xu, and Manzil Zaheer. Can public large language models help private cross-device federated learning? *arXiv preprint arXiv:2305.12132*, 2023.

[56] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems*, 33:7611–7623, 2020.

[57] Jianyu Wang, Zachary Charles, Zheng Xu, Gauri Joshi, H Brendan McMahan, Maruan Al-Shedivat, Galen Andrew, Salman Avestimehr, Katharine Daly, Deepesh Data, et al. A field guide to federated optimization. *arXiv preprint arXiv:2107.06917*, 2021.

[58] Kangkang Wang, Rajiv Mathews, Chloé Kiddon, Hubert Eichner, Françoise Beaufays, and Daniel Ramage. Federated evaluation of on-device personalization. *arXiv preprint arXiv:1910.10252*, 2019.

[59] Xiaoyang Wang, Han Zhao, Klara Nahrstedt, and Oluwasanmi O Koyejo. Robust and personalized federated learning with spurious features: an adversarial approach. 2021.

[60] Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, et al. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. *arXiv preprint arXiv:2204.07705*, 2022.

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

[62] Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning*, pages 23965–23998. PMLR, 2022.

[63] Mitchell Wortsman, Gabriel Ilharco, Jong Wook Kim, Mike Li, Simon Kornblith, Rebecca Roelofs, Raphael Gontijo Lopes, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, et al. Robust fine-tuning of zero-shot models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7959–7971, 2022.

[64] Shanshan Wu, Tian Li, Zachary Charles, Yu Xiao, Ziyu Liu, Zheng Xu, and Virginia Smith. Motley: Benchmarking heterogeneity and personalization in federated learning. In *NeurIPS Workshop on Federated Learning*, 2022.

[65] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019.

[66] Tao Yu, Eugene Bagdasaryan, and Vitaly Shmatikov. Salvaging federated learning by local adaptation. *arXiv preprint arXiv:2002.04758*, 2020.

[67] Jianyi Zhang, Saeed Vahidian, Martin Kuo, Chunyuan Li, Ruiyi Zhang, Guoyin Wang, and Yiran Chen. Towards building the federated gpt: Federated instruction tuning. *arXiv preprint arXiv:2305.05644*, 2023.

[68] Jingzhao Zhang, Sai Praneeth Karimireddy, Andreas Veit, Seungyeon Kim, Sashank Reddi, Sanjiv Kumar, and Suvrit Sra. Why are adaptive methods good for attention models? *Advances in Neural Information Processing Systems*, 33:15383–15393, 2020.

[69] Tuo Zhang, Tiantian Feng, Samiul Alam, Mi Zhang, Shrikanth S Narayanan, and Salman Avestimehr. Gpt-fl: Generative pre-trained model-assisted federated learning. *arXiv preprint arXiv:2306.02210*, 2023.

[70] Xuechen Zhang, Mingchen Li, Xiangyu Chang, Jiasi Chen, Amit K Roy-Chowdhury, Ananda Theertha Suresh, and Samet Oymak. Fedyolo: Augmenting federated learning with pretrained transformers. *arXiv preprint arXiv:2307.04905*, 2023.

[71] Haodong Zhao, Wei Du, Fangqi Li, Peixuan Li, and Gongshen Liu. Fedprompt: Communication-efficient and privacy-preserving prompt tuning in federated learning. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

A Formal Problem Setup

Federated prompt tuning. We consider a federated learning scenario consisting of n clients that communicate with a central server. For every $i \in [n]$, Client i has a dataset $\mathcal{D}_i \coloneqq \{(x_{i,j}, y_{i,j})\}_{j=1}^{m_i}$ consisting of m_i query-target pairs $(x_{i,j}, y_{i,j})$, where each query $x_{i,j}$ and target $y_{i,j}$ is a variable-length text sequence. All clients also have a copy of a language model with parameters θ, a tokenizer τ mapping text to a list of one-hot encodings of tokens, and a token embedding matrix $E \in \mathbb{R}^{e \times v}$, where e is the embedding dimension and v is the vocabulary size.

When provided an input x, the language model computes the conditional distribution of tokenized targets given the embedding of the tokenized input query, namely $\mathbb{P}_\theta(\tau(Y)|E\tau(x))$, in order to generate text predictions. A natural idea to more accurately estimate the conditional distribution of $\tau(Y)$ is to add text (a prompt) p to the input query that provides information about the relationship between inputs and targets for each task at hand, such as instructions or examples of gold-standard (x, y) pairs. In other words, the idea is that $\mathbb{P}_\theta(\tau(Y)|E\tau([p,x])) \equiv \mathbb{P}_\theta(\tau(Y)|[E\tau(p), E\tau(x)])$ should be a more accurate estimation of the true conditional distribution of Y given x for carefully chosen p. This approach is known as *in-context learning* or *prompt engineering* and has led to many successful adaptations of LLMs [2]. However, these discrete text prompts cannot be easily optimized, and restricting the embedded prompt $E\tau(p)$ to columns in E limits the information it can convey about the relationship between Y and X.

Prompt tuning [31] addresses these concerns by optimizing a "soft" prompt in embedding space. For some number of tokens k, prompt tuning aims to learn a matrix $P \in \mathbb{R}^{e \times k}$ that conditions the model for more accurate predictions when prepended to the *embedding* of the input text tokens, i.e. the new model is given by $\mathbb{P}_\theta(\tau(Y)|[P, E\tau(x)])$. In this case, the gradient of the loss of $\mathbb{P}_\theta(\tau(Y)|[P, E\tau(x)])$ with respect to P can be easily computed via backpropagation, and we can optimize P with standard gradient-based methods. This loss is the cross-entropy loss, in particular, the loss as a function of P for Client i in our federated setting is:

$$\mathcal{L}_i(P) \coloneqq -\frac{1}{m_i} \sum_{j=1}^{m_i} \log(\mathbb{P}_\theta(\tau(y_{i,j})|[P, E\tau(x_{i,j})])) \tag{1}$$

During federated training, the server aims to minimize the average loss across clients, namely $\mathcal{L}(P) \coloneqq \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}_i(P)$, and towards this end can apply standard Federated Learning algorithms such as FedAvg and FedSGD. Importantly, only the prompt embedding matrix P must be communicated between server and clients, as depicted in Figure 1.

Personalization and robustness. Due to the heterogeneity of the client datasets $\mathcal{D}_1, \ldots, \mathcal{D}_n$, the global prompt P_{glob} found by running federated learning on $\mathcal{L}(P)$ may not perform well on each client's local data. This can be addressed by personalizing P_{glob} to each client. Formally, we consider a new set of n_{test} clients with datasets $\mathcal{D}_{n+1}, \ldots, \mathcal{D}_{n+n_{\text{test}}}$ that are split into training and test sets, i.e. $\mathcal{D}_i = \mathcal{D}_i^{\text{train}} \cup \mathcal{D}_i^{\text{test}}$ for all $i = n+1, \ldots, n+n_{\text{test}}$. During personalization, Client i updates P_{glob} using its local training dataset $\mathcal{D}_i^{\text{train}}$ to obtain a prompt P_i. The level of personalization achieved by this prompt is evaluated using $\mathcal{D}_i^{\text{test}}$. However, it is also of interest to know how robust P_{glob} is to personalization, as we do not want P_i to have forgotten all of the global information it acquired during federated training. So, P_i is also evaluated on a global test dataset compiled across all client test datasets $\mathcal{D}_n^{\text{test}}, \ldots, \mathcal{D}_{n+n_{\text{test}}}^{\text{test}}$ to obtain a robustness score. These local personalization and robustness

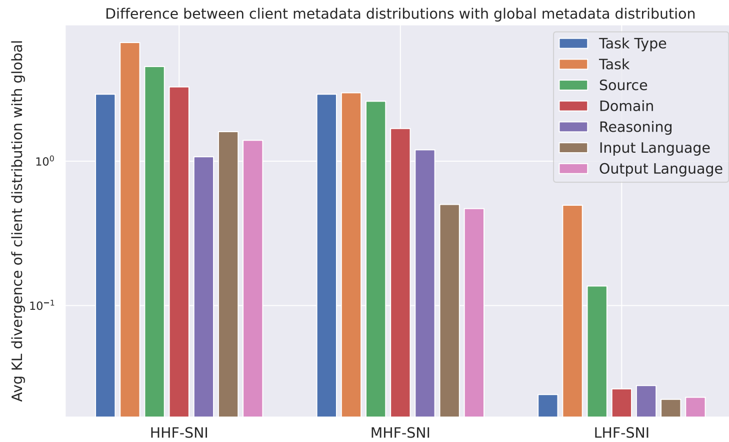

Figure 6: For each of the three training dataset partitions (HHF-SNI, MHF-SNI, LHF-SNI) and each metadata category (Task Type, Task, Source, Domain, Reasoning, Input Language, and Output Language), we plot the average across clients of the KL divergence between the client's metadata category distribution and the global metadata category distribution, in log scale.

scores are ultimately aggregated across clients and used for final evaluation of the federated algorithm used to obtain P_{glob}.

B Additional Dataset Details

Of the 76 total task types in SNI, we excluded the three type because they did not have a sufficient amount of data for one client (Punctuation Error Detection, Paper Review, Speaker Relation Classification) and one type, Mathematics, because the PaLM tokenizer cannot properly interpret numerical text input. The data was split into train/validation/test sets by randomly selecting 10% of the remaining task types each for validation and testing, and designating the rest for training. The test task types are [Irony Detection, Mathematics, Text Completion, Explanation, Overlap Extraction, Question Generation, Dialogue Act Recognition, Gender Classification] and the validation types are [Answer Verification, Information Extraction, Dialogue Generation, Commonsense Classification, Word Relation Classification, Answerability Classification, Sentence Ordering]. There are 326 total test clients and 326 total validation clients, although we only use 32 test clients, sampled uniformly from the full set of 326 test clients, in our results.

In Figure 6 we plot average Kullback-Leibler (KL) divergences between each client's meta-data distribution and the global meta-data distribution for each of our three federated partitions of SNI. The figure demonstrates that among a variety of meta-data categories, clients on average distributions of this meta-data category that differ from the global distribution to an extent that we would expect from high, medium and low-heterogeneity partitions (the larger the heterogeneity, the greater the difference between client and global distributions).

C Further Experiments and Details

Hyperparameters. In all training runs, we initialized the prompts by sampling each element i.i.d. from $\mathcal{N}(0, 0.25)$, noting that results from [31] showed that prompt initialization does not significantly affect performance at the model scale we consider ($\sim 10^{10}$ parameters). We tried prompt lengths of 5, 10, and 20, and saw that length 10 generally outperformed length 5, but there was no improvement going from length 10 to length 20, (see Figure 10) so we used length 10 for all other runs. We tuned client and server learning rates in $\{10^{-2}, 10^{-1}, 10^{0}, 10^{1}\}$ using the global validation set separately for each algorithm and each of the three training partitions, plus centralized. The resulting learning rates are found in Table 3. We tuned weight decay parameter in $\{0, 10^{-2}\}$, and Adam epsilon parameter in $\{10^{-8}, 10^{-6}, 10^{-4}\}$ on HHF-SNI and the centralized dataset, and observed that no weight decay and Adam $\epsilon = 10^{-8}$ worked best in all cases. We used $\beta_1 = 0.99$ and $\beta_2 = 0.999$ for Adam. In each trial, we used the prompt that achieved the highest global validation score during training for

Table 2: Training learning rates. All learning rates were tuned in $\{0.01, 0.1, 1, 10\}$ and chosen based on the global validation score they led to during training. The resulting values are shown here, as (server learning rate, client learning rate) if applicable. Centralized training used Adam with learning rate 1, tuned in the same set.

Algorithm	HHF-SNI	MHF-SNI	LHF-SNI
FedAvg(Adam) - prompt length 10	(1, 0.1)	(0.1, 1)	(0.1, 1)
FedAvg(SGD) - prompt length 10	(1, 10)	(0.1, 10)	(1, 10)
FedSGD - prompt length 10	1	1	1
FedSGD-LB - prompt length 10	0.01	0.1	0.1

Table 3: Adam personalization learning rates. Personalization learning rates were tuned in $\{10^{-3}, 10^{-2}, 10^{-1.5}, 10^{-1}\}$.

Algorithm	HHF-SNI	MHF-SNI	LHF-SNI
FedAvg(Adam) - High Computation	10^{-2}	10^{-2}	10^{-2}
FedAvg(Adam) - Low Computation	10^{-1}	10^{-1}	10^{-1}
FedAvg(SGD) - High Computation	10^{-2}	10^{-3}	10^{-2}
FedAvg(SGD) - Low Computation	10^{-1}	10^{-2}	10^{-1}
FedSGD - High Computation	$10^{-1.5}$	10^{-2}	10^{-2}
FedSGD - Low Computation	10^{-1}	10^{-1}	10^{-1}
FedSGD-LB - High Computation	10^{-3}	10^{-3}	10^{-3}
Centralized - High Computation	10^{-2}	10^{-2}	10^{-2}
Random-Gaussian - High Computation	10^{-2}	10^{-2}	10^{-2}
Random-Word - High Computation	10^{-2}	10^{-2}	10^{-2}

personalization. Regarding model and evaluation parameters, we set the maximum input query length to 1024 tokens and output length to 128 tokens for training and 10 tokens for evaluation, and the decoding temperature to 0, following [60]. For examples with multiple targets, we take the max score over targets, again following [60].

C.1 Additional results

In this section we provide additional empirical results. Unless otherwise noted, all experiments run personalization with Adam on a dataset of size 256.

Role of personalization learning rate with FedSGD-trained prompts. In Figure 7 we verify that using a smaller personalization learning rate improves the personalization-robustness trade-off for FedSGD-trained prompts, just like we observed for FedAvg(Adam)-trained prompts in Figure 3(Left). Again, increased robustness (higher global scores) comes at the cost of additional personalization epochs required to reach high local scores.

Variation across training runs. In Figures 8 and 9 we plot versions of Figure 4 with different random seeds for training. In each case the takeaway is the same as Observations 3a,b: FedAvg(Adam) outperforms FedAvg(SGD), and FedAvg(Adam) generally outperforms FedSGD, especially when trained on low-heterogeneity data and especially in terms of global scores. The one case in which FedSGD yields a better personalization-robustness tradeoff is on HHF-SNI (high heterogeneity) with seed 0 (Figure 8) due to higher local scores for FedSGD.

SGD as personalization optimizer. One may suspect that the improvement of FedAvg(Adam) over FedAvg(SGD) in the previous results is due to FedAvg(Adam) using the same client optimizer as the personalization optimizer (Adam). However, Figure 10 we show that the relative performance of FedAvg(Adam) and FedAvg(SGD) does not change when SGD is used as the personalization optimizer rather than Adam.

Impact of fewer personalization samples. In Figure 11 we plot results from personalization with varying number of of examples per client, namely 64 and 256. With only 64 samples, late in training overfitting to the training set occurs to extent that even local scores decrease. Further, the best local

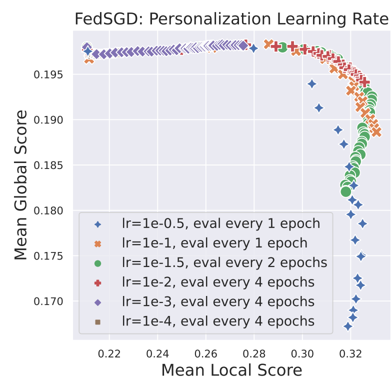

Figure 7: Mean global and local scores across test clients during personalization with varying learning rates from a prompt trained on HHF-SNI by **FedSGD**. All runs besides those with the largest two learning rates are run for 100 epochs, and otherwise 20 epochs.

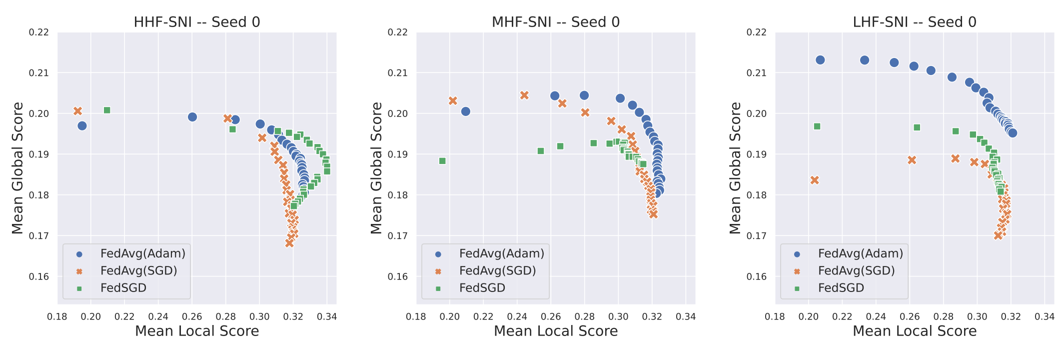

Figure 8: Version of Figure 4 with random seed 0. Mean global and local scores across test clients evaluated every 4 epochs during **100 epochs** of personalization (High Computation regime) starting from prompts pre-trained by FedAvg(Adam), FedAvg(SGD) and FedSGD with random seed 0 on **(Left)** HHF-SNI, **(Center)** MHF-SNI, and **(Right)** LHF-SNI.

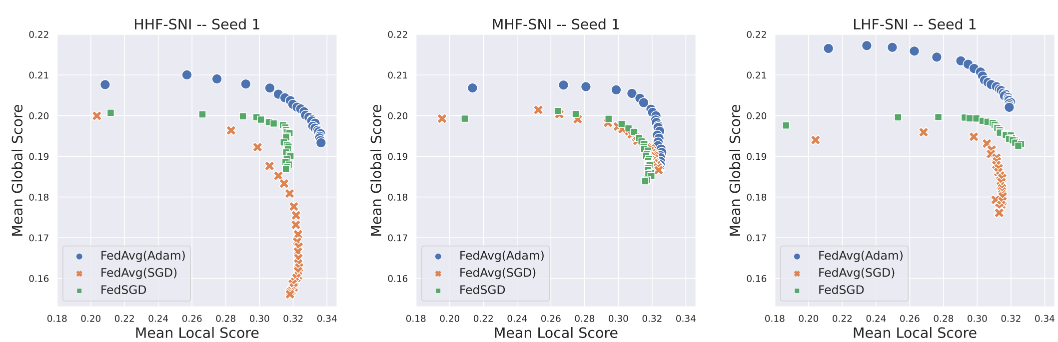

Figure 9: Version of Figure 4 with random seed 1. Mean global and local scores across test clients evaluated every 4 epochs during **100 epochs** of personalization (High Computation regime) starting from prompts pre-trained by FedAvg(Adam), FedAvg(SGD) and FedSGD with random seed 1 on **(Left)** HHF-SNI, **(Center)** MHF-SNI, and **(Right)** LHF-SNI.

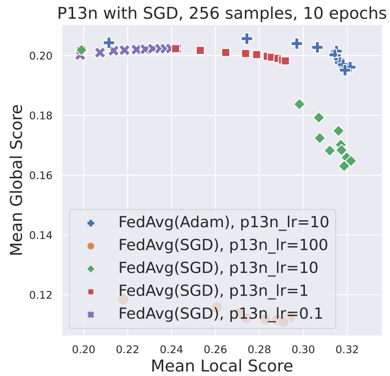

Figure 10: Personalization with SGD. Mean global and local scores across test clients evaluated every epoch during 10 epochs of personalization with SGD, starting from prompts pre-trained by FedAvg(Adam) and FedAvg(SGD) on HHF-SNI.

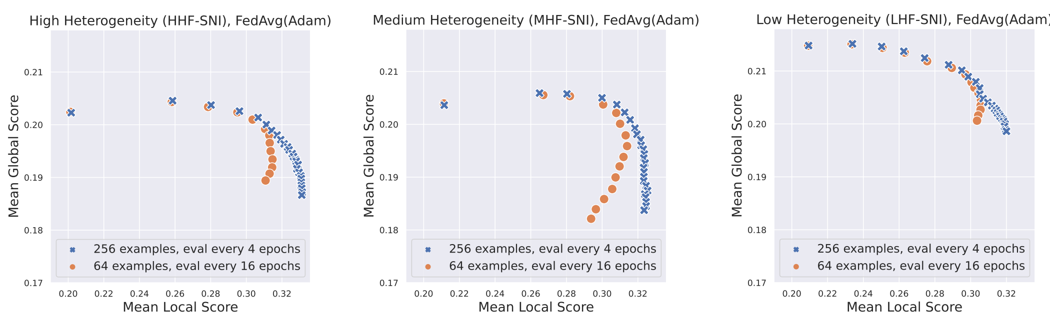

Figure 11: Impact of fewer personalization instances. Global and local scores during personalization on either 256 or 64 instances (examples) starting from prompts pre-trained on **(left)** HHF-SNI, **(center)** MHF-SNI, and **(right)** LHF-SNI. For 256 instances, 100 epochs are executed, and for 64 instances, 224 epochs are executed (High Computation regime).

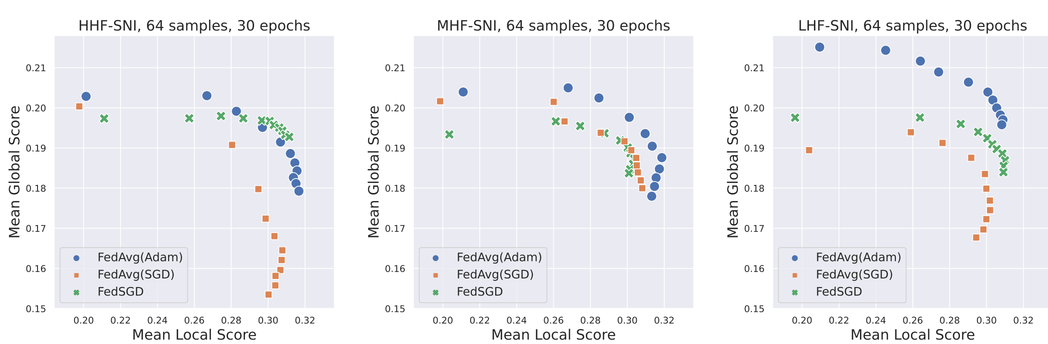

Figure 12: Low computation, 64 instances. Mean global and local scores across test clients evaluated every 3 epochs during personalization with 30 total epochs of 64 instances (samples) per epoch, from prompts pre-trained on **(left)** HHF-SNI, **(center)** MHF-SNI, and **(right)** LHF-SNI.

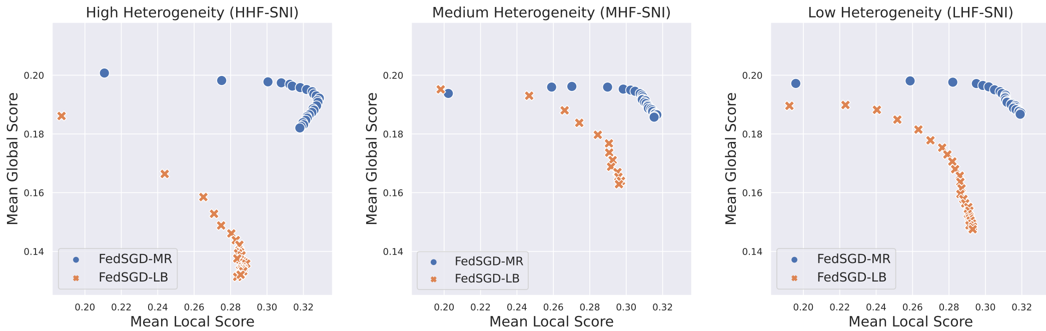

Figure 13: FedSGD with many rounds vs large batch size. Mean global and local scores across test clients during personalization starting from prompts pre-trained by FedSGD with many rounds (FedSGD-MR, referred to as FedSGD in all other experiments) and FedSGD with large batch size (FedSGD-LB) on **(left)** HHF-SNI, **(center)** MHF-SNI, and **(right)** LHF-SNI.

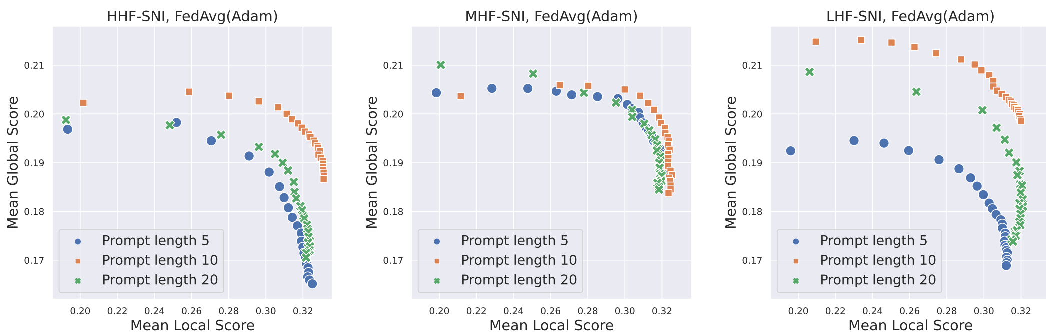

Figure 14: Role of prompt length – FedAvg(Adam). Mean global and local scores evalutated every 4 epochs during 100 epochs of personalization on 256 instances starting from prompts of varying lengths pre-trained by FedAvg(Adam) on **(left)** HHF-SNI, **(center)** MHF-SNI, and **(right)** LHF-SNI.

score for 64 examples is smaller than the best local score for 256 examples by about 0.01-0.02 for each heterogeneity level. However, fewer local samples reduces local scores more so than global scores, and early in training the personalization-robustness trade-off is roughly equivalent to that with 256 examples.

In Figure 12, we compare the personalization vs robustness trade-off for FedAvg(Adam), FedAvg(SGD), and FedSGD-trained prompts with few instances (64) in the Low Computation rage (30 epochs). Note that this is more updates than the previously studied Low Computation cases, which ran for 10 epochs, but the total amount of computation is actually less because we are here running epochs of 64 instances rather than 256 instances in the previous case. The relative ordering of performance among the three FL algorithms stays the same, with the exception of FedSGD arguably slightly outperforming FedAVg(Adam) in the heterogeneity case.

Variants of FedSGD. In all previous experiments we have used the version of FedSGD that has the same client batch size (32) and number of active clients per round (32) as the FedAvg variants we experiment with, but executes 16x more communication rounds than the FedAvg variants (4800 rounds vs 1600 rounds) so that it sees the same total number of instances (since the FedAvg variants make 16 local updates per client per round, whereas FedSGD makes effectively only 1). Now, we experiment with a different version of FedSGD that multiplies the client batch size by 16 rather than the number of communication rounds. In particular, this version, which we call **FedSGD-L**arge**B**atch, uses a client batch size of 512, and samples 32 clients per round for 300 rounds. Like the other FL algorithms, it uses Adam as its server optimizer. Figure 13 shows that the original version of FedSGD with **m**any **r**ounds (referred to here as FedSGD-MR) far outperforms FedSGD-LB, implying that it is advantageous to do more updates with noisier gradients.rather thank fewer updates with less noisy gradients.

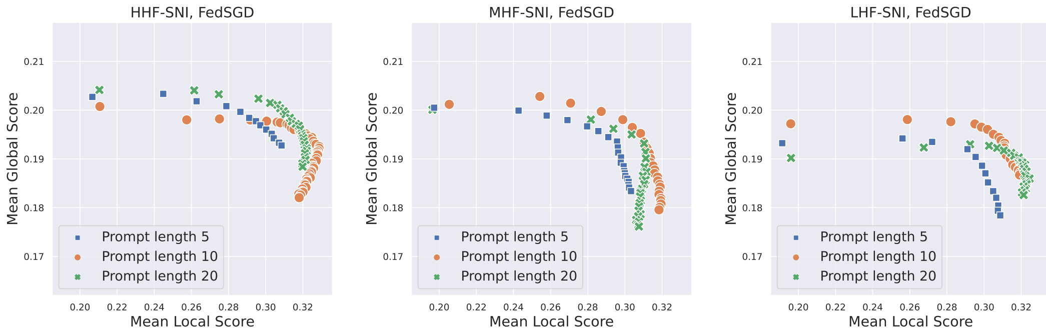

Figure 15: Role of prompt length – FedSGD. Mean global and local scores evalutated every 4 epochs during 100 epochs of personalization on 256 instances starting from prompts of varying lengths pre-trained by FedSGD on **(left)** HHF-SNI, **(center)** MHF-SNI, and **(right)** LHF-SNI.

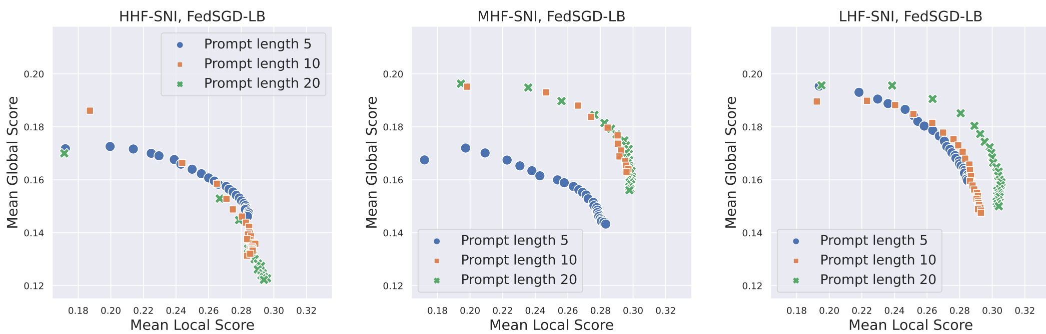

Figure 16: Role of prompt length – FedSGD-LB. Mean global and local scores evalutated every 4 epochs during 100 epochs of personalization on 256 instances starting from prompts of varying lengths pre-trained by FedSGD-LB on **(left)** HHF-SNI, **(center)** MHF-SNI, and **(right)** LHF-SNI.

Role of prompt length. In Figures 14, 15 and 16 we explore the effect of changing the prompt length for FedAvg(Adam), FedSGD and FedSGD-LB, respectively, in the High Computation personalization regime with 100 epochs of 256 samples. Prompt length 10 seems to be the sweet spot, as prompt length 5 gives the worst personalization vs robustness trade-off in all cases besides FedSGD-LB on HHF-SNI, and prompt length 20 provides clear improvement over prompt length 10 only in one case (FedSGD-LB on LHF-SNI), and can sometimes do significantly worse (as in the FedAvg(Adam) cases). The takeaway is similar to that in [31]: increasing the number of tokens in soft prompts improves performance up to some number of tokens, but beyond this there is no benefit to further increasing the prompt length.

Variation in client performance. Thus far all of our results have been mean scores across 32 test clients. Now, we investigate the variation in performance across clients. In Figure 17, we plot each of the 32 test clients' scores pre- and post-personalization in the Low Computation regime with 10 epochs of personalization on 256 instances, starting from prompts trained by FedAvg(Adam) on HHF-SNI. With the exception of one outlying client, the width of the range of local scores is roughly equivalent before and after personalization, while there is a large variance in global scores post-personalization.

In Figure 18, we plot 90th and 10th percentile client global and local scores during personalization in the High Computation regime with 100 epochs of 256 instances from prompts trained by FedAvg(Adam), FedAvg(SGD), and FedSGD. That is, instead of each point representing (mean local score, mean global score) across clients during some personalization epoch, they instead represent (90th percentile local score, 90th percentile global score) across clients during some personalization epoch (and likewise for the 10th percentile). This yields a number of takeaways: 1) The worst local scores are roughly the same for all algorithms and during all personalization epochs, indicating that there are some very hard clients; 2) for all algorithms, the worst global scores drop significantly during

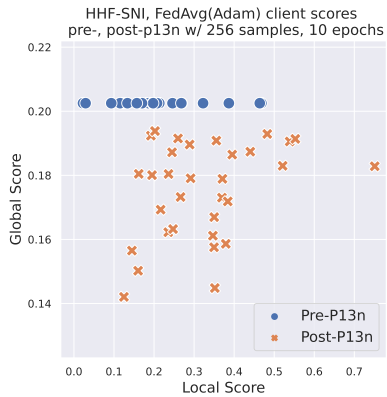

Figure 17: Per-client global and local scores before and after personalization (p13n) consisting of 10 epochs on 256 examples from prompts pre-trained by FedAvg(Adam) on HHF-SNI.

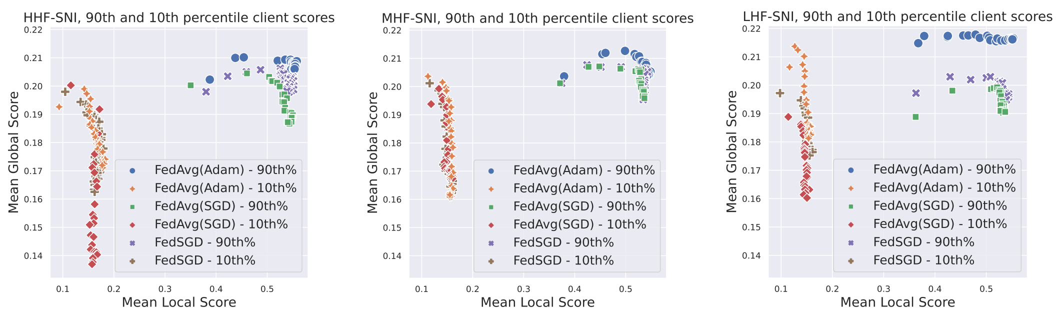

Figure 18: Global and local score 90th and 10th percentiles across test clients during personalization with 100 epochs of 256 instances from prompts pre-trained on **(left)** HHF-SNI, **(center)** MHF-SNI, and **(right)** LHF-SNI. Scores are evaluated every 4 epochs.

personalization; 3) in contrast, the best global scores do not change much during personalization, and the best local scores increase significantly.

C.2 Personalization heuristics

In this Section we first describe in greater detail the heuristics evaluated in Figure 5, then explore additional heuristics in Figure 19.

l2 regularization. Let P_{glob} be the global prompt resulting from federated training, P_i be the prompt the Client i personalizes, and $P_{i,10}$ be the prompt resulting from 10 epochs of personalization to Client i. The first heuristic we consider, l2 regularization, adds the regularizer

$$\frac{\lambda}{2}\|P_i - P_{\text{glob}}\|_F^2$$

to the loss for Client i, then runs personalization as usual. This encourages P_i to stay close to P_{glob} during personalization, which should reduce forgetting.

Model averaging. Model averaging, first runs personalization to completion, then computes interpolated prompts:

$$P_{i,10*\alpha} = \alpha P_{i,10} + (1-\alpha)P_{\text{glob}}$$

for $\alpha \in \{0, 0.1, 0.2, ..., 1\}$. Each plotted point in Figure 5 corresponds to the average local and global scores for $P_{i,10*\alpha}$ across all clients $i \in [32]$ for a particular value of α.

Note that l2 regularization are orthogonal and can be combined. We do this in Figure 5 for the scores with label "Model Averaging, $\lambda = 10^{-3}$".

Additional results. Figure 19 shows the same results as Figure 5 plus results for three additional personalization approaches:

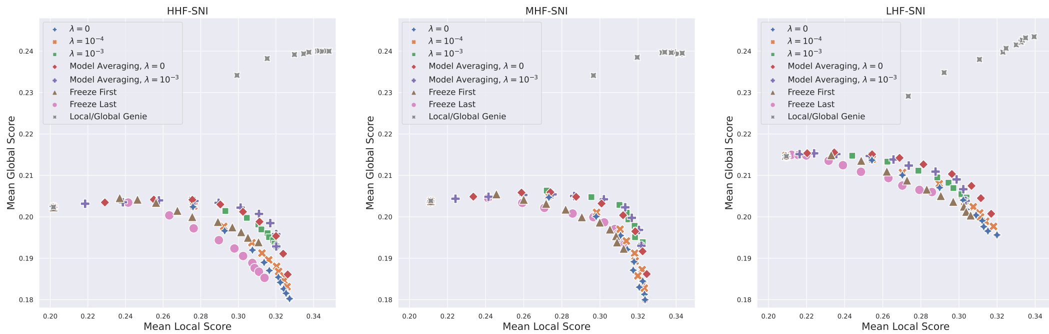

Figure 19: Personalization heuristics – Low Computation. Mean local and global scores during 10 epochs of personalization with various heuristics starting from prompts trained by FedAvg(Adam) on **(Left)** HHF-SNI, **(Center)** MHF-SNI, and **(Right)** LHF-SNI.

- **Freeze First.** Recall that P is a matrix of size prompt length (in tokens) by embedding dimension, where here the prompt length is 10. For "Freeze First", we freeze the first 8 rows (tokens) and only update the last two rows of P_i (starting from P_{glob}) during personalization.
- **Freeze Last.** Likewise, for "Freeze Last", we only update the first two rows of P_i. Neither "Freeze First" nor "Freeze Last" confer any improvement to the personalization-robustness trade-off.
- **Local/Global Genie.** These scores are the scores of a genie that knows the whether the personalized or global prompt will result in a prediction with larger score for a particular input and target, and uses the prompt with higher score for that input. It is equivalent to running inference twice for every input, once with the personalized prompt and once with the global prompt, and recording the max score among the two predictions, given the target. This is not a realistic personalization method because in practice the target is unknown. Nevertheless, we find it to be a valuable measure of the combined capabilities of personalized and global prompts, i.e. the combined information between the personalized and global prompts. The very strong performance of this genie suggests that personalization-robustness trade-offs can be drastically improved by appropriately selecting whther to use the personalized prompt or global prompt for every input query (in fact, there would no longer be a trade-off – both personalized and global scores would inrease). To train the personalized prompt, we we run vanilla personalization (i.e. $\lambda = 0$ in Figure 19).