Leveraging Side Information for Communication-Efficient Federated Learning

Berivan Isik [*1] **Francesco Pase** [*2] **Deniz Gunduz** [3] **Sanmi Koyejo** [1] **Tsachy Weissman** [1] **Michele Zorzi** [2]

Abstract

The high communication cost of sending model updates from the clients to the server is a significant bottleneck for scalable federated learning (FL). Among existing approaches, state-of-the-art bitrate-accuracy tradeoffs have been achieved using stochastic compression methods – in which the client n sends a sample from a client-only probability distribution $q_{\phi^{(n)}}$, and the server estimates the mean of the clients' distributions using these samples. However, such methods do not take full advantage of the FL setup where the server, throughout the training process, has *side information* in the form of a pre-data distribution p_θ that is close to the client's distribution $q_{\phi^{(n)}}$ *in Kullback–Leibler (KL) divergence*. We exploit this *closeness* between the clients' distributions $q_{\phi^{(n)}}$'s and the side information p_θ at the server, and propose a framework that requires approximately $D_{KL}(q_{\phi^{(n)}}||p_\theta)$ bits of communication. We show that our method can be integrated into many existing stochastic compression frameworks such as `FedPM`, Federated `SGLD`, and `QSGD` to attain the same (and often higher) test accuracy with up to 50 times reduction in the bitrate. (See (Isik et al., 2023a) for the full version.)

1. Introduction

Federated learning (FL), while enabling model training without collecting clients' raw data, suffers from high communication costs due to the model updates communicated from the clients to the server every round (Kairouz et al., 2021). To mitigate this cost, several communication-efficient FL strategies have been developed that compress the model updates, such as sparsification (Barnes et al., 2020; Isik et al., 2022; Lin et al., 2018), quantization (Alistarh et al., 2017; Mitchell et al., 2022), low-rank factorization (Basat et al., 2022; Konečnỳ et al., 2016; Vargaftik et al., 2022; 2021), and sparse network training (Isik et al., 2023b). Many of these strategies adopt a stochastic approach that requires the client n to send a sample $\mathbf{x}^{(t,n)}$ from a client-only distribution $q_{\phi^{(t,n)}}$ (which we call the *post-data distribution*), while the goal of the server is to estimate $\mathbb{E}_{X^{(t,n)}\sim q_{\phi^{(t,n)}},\forall n\in[N]}\left[\frac{1}{N}\sum_{n=1}^{N} X^{(t,n)}\right]$ by taking the average of the samples across clients $\frac{1}{N}\sum_{n=1}^{N}\mathbf{x}^{(t,n)}$. Here, we denote by N the number of clients, by $[N]$ the set $\{1,\dots,N\}$, and by $\mathbf{a}_i^{(t,n)}$ the i-th parameter of a vector $\mathbf{a}$ at client n in round t. We show that in many stochastic FL settings, the server also holds a distribution $p_{\theta^{(t)}}$ (which we call the *pre-data distribution*) that is close to the post-data distribution $q_{\phi^{(t,n)}}$ (which is unknown to the server) in KL divergence. The proposed method, KL Minimization with Side Information (`KLMS`), exploits this closeness to reduce the cost of communicating samples $\mathbf{x}^{(t,n)}$. We refer the reader to Appendix A.1 for a summary of three such stochastic FL frameworks with pointers to the corresponding pre-data $p_{\theta^{(t)}}$ and post-data $q_{\phi^{(t,n)}}$ distributions as examples of three setups: (i) learning probability distributions over subnetworks (or masks), (ii) learning deterministic model parameters using stochastic compressors, and (iii) learning probability distributions over model parameters.

Before describing the details of our proposal in Section 2, we briefly give the key idea `KLMS` relies on: Instead of communicating the deterministic value of a sample $\mathbf{x}^{(t,n)}\sim q_{\phi^{(t,n)}}$, client n can communicate a sample $\mathbf{y}^{(t,n)}$ from another distribution $\mathbf{y}^{(t,n)}\sim\tilde{q}_{\pi^{(t,n)}}$, which is less costly to communicate compared to $\mathbf{x}^{(t,n)}$, and the discrepancy due to sampling from this distribution is not significant. As shown in Algorithm 1, to construct $\tilde{q}_\pi$, we use the pre-data distribution $p_{\theta^{(t)}}$ (which is known by the server and the clients) and the importance sampling algorithm in (Chatterjee & Diaconis, 2018). We show that this `KLMS` an arbitrarily small discrepancy in the estimation when $K\simeq\exp\left(D_{KL}(q_{\phi^{(n)}}\|p_\theta)\right)$ with improvements (specific to the FL setting) over prior work (Havasi et al., 2019; Triastcyn et al., 2021). Clearly, to get the most communication gain out of `KLMS`, we need pre-data p_θ and post-data $q_{\phi^{(n)}}$ distributions that are close in KL divergence. We show the

[*]Equal contribution [1]Stanford University [2]University of Padova [3]Imperial College London. Correspondence to: Berivan Isik <berivan.isik@stanford.edu>, Francesco Pase <pasefrance@dei.unipd.it>.

ICML 2023 Workshop on Federated Learning and Analytics in Practice: Algorithms, Systems, Applications, and Opportunities. This workshop does not have official proceedings and this paper is non-archival.

existence of such distributions in many stochastic FL frameworks by providing concrete examples in Appendix A.1.

Algorithm 1 KLMS Outline. (A more detailed description is given in Section 2.1 and Appendix B.)

(1) The server and client n generate the *same* K samples from the pre-data distribution $\{\mathbf{y}_{[k]}^{(t,n)}\}_{k=1}^{K} \sim p_{\theta^{(t)}}$ (which is available to both the server and the clients) using a shared random seed.
(2) Client n computes the importance weights $\alpha_{[k]} = \frac{q_{\phi^{(t,n)}}(\mathbf{y}_{[k]}^{(t,n)})}{p_{\theta^{(t)}}(\mathbf{y}_{[k]}^{(t,n)})}$ for $k \in [K]$ with the local post-data distribution $q_{\phi^{(t,n)}}$ and normalizes it to get a distribution over $[K]$ as $\pi^{(t,n)}(k) = \frac{\alpha_{[k]}}{\sum_{l=1}^{K}\alpha_{[l]}}$.
(3) Client n takes a sample from this new distribution $k^{(n)*} \sim \pi^{(t,n)}$ and sends it to the server in $\log K$ bits.
(4) The server receives $k^{(n)*}$ and recovers the $k^{(n)*}$-th sample $\mathbf{y}_{[k^{(n)*}]}^{(t,n)}$ from the set of K samples $\{\mathbf{y}_{[k]}^{(t,n)}\}_{k=1}^{K}$ generated from $p_{\theta^{(t)}}$ in Step (1). Notice that $\mathbf{y}_{[k^{(n)*}]}^{(t,n)}$ is actually a sample from the underlying distribution over $\{\mathbf{y}_{[k]}^{(t,n)}\}_{k=1}^{K}$ defined as $\tilde{q}_{\pi^{(t,n)}}(\mathbf{y}) = \sum_{k=1}^{K}\pi^{(t,n)}(k)\cdot\mathbf{1}\{\mathbf{y}_{[k]}^{(t,n)} = \mathbf{y}\}$.

Each of the three examples of stochastic communication-efficient FL frameworks listed in Appendix A.1, induces a post-data distribution $q_{\theta^{(t,n)}}$ that clients want to send a sample from, and a pre-data distribution $p_{\theta^{(t)}}$ that is available to both the clients and the server – playing the role of side information. In each case, these distributions are expected to become closer in KL divergence as training progresses due to the convergence of the model parameters (FedPM (Isik et al., 2023b) or other probabilistic mask learning methods), temporal correlation across rounds (QSGD (Alistarh et al., 2017) or other deterministic model training methods), or the stochastic formulation of the framework itself (Federated SGLD (Vono et al., 2022) or other Bayesian FL methods). We show that KLMS reduces the communication cost down to this *fundamental quantity* (KL divergence) in each scenario, resulting in up to 50 times improvement in communication efficiency (sometimes with higher accuracies) over FedPM, QLSD, and QSGD among other non-stochastic competitive baselines such as SignSGD (Bernstein et al., 2018), TernGrad (Wen et al., 2017), DRIVE (Vargaftik et al., 2021), EDEN (Vargaftik et al., 2022), and FedMask (Li et al., 2021). To achieve this efficiency, we use an importance sampling algorithm (Chatterjee & Diaconis, 2018; Harsha et al., 2007) by improving and extending the previous theoretical guarantees to the distributed setting. Different from prior work that used importance sampling in the centralized setting to compress model parameters (Havasi et al., 2019) or focused on differential privacy implications (Shah et al., 2022; Triastcyn et al., 2021), KLMS selects more natural pre-data $p_{\theta^{(t)}}$ and post-data $q_{\phi^{(t,n)}}$ distributions that are intrinsic to the FL setting, and optimizes the bit allocation across both the training rounds and the model coordinates in an adaptive way to achieve the optimal bitrate. Our contributions can be summarized as follows:

(1) We propose a road map to utilize various forms of side information available to both the server and the clients to reduce the communication cost in FL. We give concrete examples of how to code model updates under different setups, including probabilistic mask training (e.g., FedPM), deterministic model training with stochastic compressors (e.g., QSGD), and Bayesian FL (e.g., Federated SGLD).

(2) We extend the importance sampling results to the distributed setting with theoretical improvements.

(3) We propose an adaptive bit allocation strategy that eliminates a hyperparameter required by prior work, and allows a better use of the communication budget across the model coordinates and rounds.

(4) We demonstrate the efficacy of our strategy on MNIST, EMNIST, CIFAR-10, and CIFAR-100, and show improvements in accuracy with up to 50 times gains in bitrate (with sometimes higher accuracies) over relevant baselines.

2. KL Divergence Minimization with Side Information (KLMS)

We first describe our approach, KLMS, in Section 2.1; then, in Section 2.2, we introduce our adaptive bit allocation strategy to optimize the bitrate across training rounds and model coordinates to reduce the compression rate. In Appendix D, we give four concrete examples where KLMS is integrated into FedPM (Isik et al., 2023b), QSGD (Alistarh et al., 2017), SignSGD (Bernstein et al., 2018), and Federated SGLD (Vono et al., 2022); and improves the accuracy-bitrate tradeoff.

2.1. KLMS for Stochastic FL Frameworks

We first point out that our proposal is not a stand-alone FL framework to replace existing alternatives, rather, it represents a general recipe that can be integrated into many existing (stochastic) frameworks to improve their accuracy-bitrate performance significantly. The main idea behind KLMS is grounded in three important observations:

(1) In many existing FL frameworks, the updates communicated from the clients to the server are samples drawn from some optimized post-data distributions, e.g., QSGD (Alistarh et al., 2017) and FedPM (Isik et al., 2023b).

(2) Sending a *random* sample from a distribution can be done much more efficiently than first taking a sample from the same distribution, and then sending its *deterministic* value (Theis & Ahmed, 2022).

(3) The knowledge acquired from the historical updates, available both to the server and clients, can reduce the communication cost drastically by acting as side information.

KLMS is designed to reduce the communication cost in FL by taking advantage of the above observations. It relies on common randomness between the clients and the server in the form of a random SEED (i.e., they can generate the same random samples from the same distribution) and also on the side information available to the server and the clients. Without restricting ourselves to any specific FL framework (we do this in Appendix D), suppose the server and the clients share a pre-data distribution $p_{\theta^{(t)}}$, and each client has a post-data distribution $q_{\phi^{(t,n)}}$ after the local training steps. As stated in Section 1, the goal of the server is to compute $\mathbb{E}_{X^{(t,n)}\sim q_{\phi^{(t,n)}},\forall n\in[N]}\left[\frac{1}{N}\sum_{n=1}^{N}X^{(t,n)}\right]$ after each round. While this can be done by simply communicating samples $\mathbf{x}^{(t,n)}\sim q_{\phi^{(t,n)}}$, we note that the communicated samples do not need to be the exact same samples that are generated at the client's side. Therefore, instead of communicating a specific realization $\mathbf{x}^{(t,n)}\sim q_{\phi^{(t,n)}}$, KLMS communicates a sample $\mathbf{y}^{(t,n)}$ according to some other distribution $\tilde{q}_{\pi^{(t,n)}}$ such that (i) it is less costly to communicate a sample from $\tilde{q}_{\pi^{(t,n)}}$ rather than $q_{\phi^{(t,n)}}$, and (ii) the discrepancy $E=\left|\mathbb{E}_{Y^{(t,n)}\sim\tilde{q}_{\pi^{(t,n)}},\forall n\in[N]}\left[\frac{1}{N}\sum_{n=1}^{N}Y^{(t,n)}\right]-\mathbb{E}_{X^{(t,n)}\sim q_{\phi^{(t,n)}},\forall n\in[N]}\left[\frac{1}{N}\sum_{n=1}^{N}X^{(t,n)}\right]\right|$ is sufficiently small. Motivated by this, each round of KLMS runs as described in Algorithm 1. Theorem 2.1 shows that the discrepancy E is upper bounded when $K\simeq\exp\left(D_{KL}(q_\phi||p_\theta)\right)$. We prove it for a general measurable function $f(\cdot)$, for which the discrepancy E is a special case when $f(\cdot)$ is the identity. We note that the previous results on the single-user scenario ($N=1$) (Chatterjee & Diaconis, 2018) are special cases of our more general framework with N users.

Theorem 2.1. *Let p_θ and $q_{\phi^{(n)}}$ for $n=1,\dots,N$ be probability distributions over set $\mathcal{X}$ equipped with some sigma-algebra. Let $X^{(n)}$ be an $\mathcal{X}$-valued random variable with law $q_{\phi^{(n)}}$. Let $r\geq 0$ and $\tilde{q}_{\pi^{(n)}}$ for $n=1,\dots,N$ be discrete distributions each constructed by $K^{(n)}=\exp\left(D_{KL}(q_{\phi^{(n)}}||p_\theta)+r\right)$ samples $\{\mathbf{y}_{[k]}^{(n)}\}_{k=1}^{K^{(n)}}$ from p_θ defining $\pi^{(n)}(k)=\frac{q_{\phi^{(n)}}(\mathbf{y}_{[k]}^{(n)})/p_\theta(\mathbf{y}_{[k]}^{(n)})}{\sum_{l=1}^{K^{(n)}}q_{\phi^{(n)}}(\mathbf{y}_{[l]}^{(n)})/p_\theta(\mathbf{y}_{[l]}^{(n)})}$. Furthermore, for measurable function $f(\cdot)$, let $\|f\|_{\mathbf{q}_\phi}=\sqrt{\mathbb{E}_{X^{(n)}\sim q_{\phi^{(n)}},\forall n\in[N]}[(\frac{1}{N}\sum_{n=1}^{N}f(X^{(n)}))^2]}$ be its 2-norm under $\mathbf{q}_\phi=q_{\phi^{(1)}},\dots,q_{\phi^{(N)}}$ and let*

$$\epsilon=\left(e^{-Nr/4}+2\sqrt{\prod_{n=1}^{N}\mathbb{P}(\log(q_{\phi^{(n)}}/p_\theta)>D_{KL}(q_{\phi^{(n)}}\|p_\theta)+r/2)}\right)^{1/2}. \tag{1}$$

Defining $\tilde{q}_{\pi^{(n)}}$ over $\{\mathbf{y}_{[k]}^{(n)}\}_{k=1}^{K^{(n)}}$ as $\tilde{q}_{\pi^{(n)}}(\mathbf{y})=\sum_{k=1}^{K^{(n)}}\pi^{(n)}(k)\cdot\mathbf{1}(\mathbf{y}_{[k]}^{(n)}=\mathbf{y})$, it holds that

$$\mathbb{P}\left(\left|\mathop{\mathbb{E}}_{Y^{(n)}\sim\tilde{q}_{\pi^{(n)}},\forall n}\left[\frac{1}{N}\sum_{n=1}^{N}f(Y^{(n)})\right]-\mathop{\mathbb{E}}_{X^{(n)}\sim q_{\phi^{(n)}},\forall n}\left[\frac{1}{N}\sum_{n=1}^{N}f(X^{(n)})\right]\right|\geq\frac{2\|f\|_{\mathbf{q}_\phi}\epsilon}{1-\epsilon}\right)\leq 2\epsilon,$$

See Appendix C for the proof. This result implies that when $K^{(n)}\simeq\exp\left(D_{KL}(q_{\phi^{(t,n)}}\|p_{\theta^{(t)}})\right)$, the discrepancy E is small. In practice, as we explain in Section 2.2, we work on blocks of parameters such that $D_{KL}(q_{\phi^{(t,n)}}\|p_{\theta^{(t)}})$ for each block is the same for all clients $n\in[N]$. Hence, we omit the superscript (n) from $K^{(n)}$.

2.2. Adaptive Block Selection for Optimal Bit Allocation

Prior work has applied importance sampling for Bayesian neural network compression (Havasi et al., 2019), or for differentially private communication in FL (Triastcyn et al., 2021) by splitting the model into several fixed-size blocks of parameters, and compress each block separately and independently to avoid the high computational cost – which exponentially increases with the number of parameters d. After splitting the model into fixed-size blocks with S parameters each, (Havasi et al., 2019; Triastcyn et al., 2021) choose a single fixed K (number of samples generated from $p_\theta^{(t)}$) for each block no matter what the KL divergence is for different blocks. This yields the same bitrate $\frac{\log K}{S}$ for every model parameter. Furthermore, (Triastcyn et al., 2021) uses the same K throughout training without considering the variation in KL divergence over rounds. However, as illustrated in Figure 2, KL divergence changes significantly across different layers and rounds. Hence, spending the same bitrate $\frac{\log K}{S}$ for every parameter every round is highly suboptimal since it breaks the condition in Theorem 2.1.

To fix this, we propose an adaptive block selection mechanism, where the block size is adjusted such that the KL divergence for each block is the same and equal to a target value, D_{KL}^{target}. This way, the optimal K for each block is the same and approximately equal to D_{KL}^{target}, and we do not need to set the block size S ourselves, which was a hyperparameter to tune in (Havasi et al., 2019; Triastcyn et al., 2021). Different from the fixed-size block selection approach in (Havasi et al., 2019; Triastcyn et al., 2021), the adaptive approach requires describing the locations of the adaptive-size blocks, which adds overhead to the communication cost. However, exploiting the temporal correlation across rounds can make this overhead negligible. More specifically, we first let each client find their adaptive-size blocks, each having KL divergence equal to D_{KL}^{target}, in the first round. Then the clients communicate the locations of these blocks to the server, which are then aggregated by the

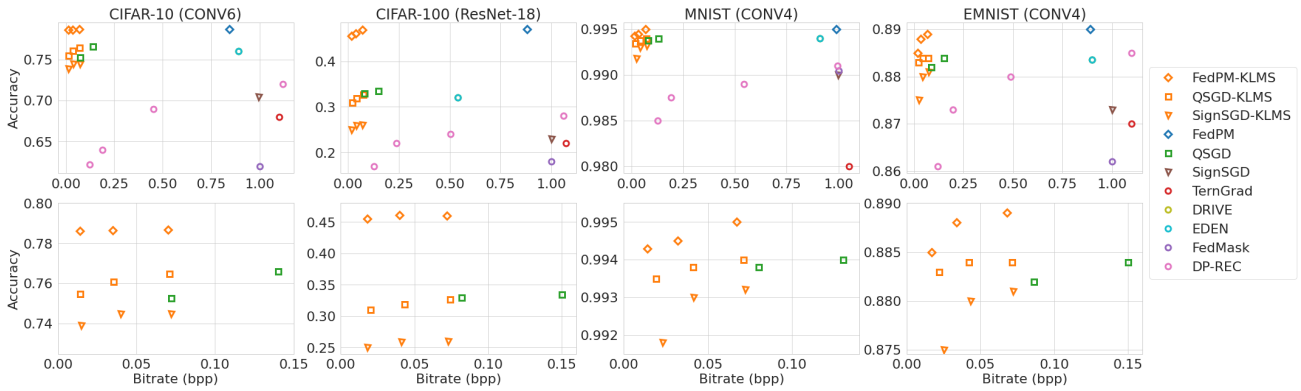

Figure 1: FedPM−KLMS, QSGD−KLMS, and SignSGD−KLMS against FedPM, QSGD, SignSGD, TernGrad, DRIVE, EDEN, FedMask, and DP−REC. The bottom row replicates the upper row zoomed into lower bitrates.

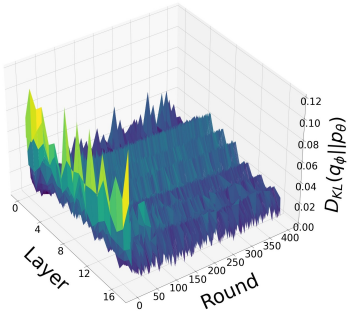

Figure 2: KL divergence between local post-data distributions and the global pre-data distribution, for different layers and rounds (FedPM is used to train CONV6 on CIFAR-10).

server to find the new global indices to be broadcast to the clients, i.e., federated aggregation of block locations. At later rounds, the server checks if, on average, the new KL divergence of the previous blocks is still sufficiently close to the target value D_{KL}^{target}. If so, the same adaptive-size blocks are used in that round. Otherwise, the client constructs new blocks, each having KL divergence equal to D_{KL}^{target}, and updates the server about the new locations. Our experiments indicate that this update occurs only a few times during the whole training. Therefore, it adds only a negligible overhead on the average communication cost across rounds. We provide the pseudocodes in Appendix B.

3. Experiments

We empirically demonstrate the accuracy and bitrate improvements obtained with KLMS by focusing on four KLMS adaptations, FedPM−KLMS, QSGD−KLMS, and SignSGD−KLMS, and SGLD−KLMS, covered in Appendix D. Due to the page limitation, we only provide the results of non-Bayesian FL setup with i.i.d. data split and full participation here. For the non-i.i.d. data split, partial client participation, and Bayesian FL experiments, please see Appendix F. We also provide an ablation study on the effectiveness of the adaptive block selection strategy in Appendix F.3. In this section though, we consider four datasets: CIFAR-10 (Krizhevsky et al., 2009), CIFAR-100 (Krizhevsky et al., 2009), MNIST (Deng, 2012), and EMNIST (Cohen et al., 2017) (with 47 classes). For CIFAR-100, we use ResNet-18 (He et al., 2016); for CIFAR-10, a 6-layer CNN CONV6; for MNIST a 4-layer CNN CONV4; and for EMNIST, again CONV4. Clients perform 3 local epochs in the non-Bayesian. (Results averaged over 3 runs.) In Figure 1, we compare FedPM−KLMS, QSGD−KLMS, and SignSGD−KLMS with FedPM (Isik et al., 2023b), QSGD (Alistarh et al., 2017), SignSGD (Bernstein et al., 2018), TernGrad (Wen et al., 2017), DRIVE (Vargaftik et al., 2021), EDEN (Vargaftik et al., 2022), FedMask (Li et al., 2021), and DP−REC (Triastcyn et al., 2021). It is seen that FedPM−KLMS and SignSGD−KLMS provide 50 times reduction in communication cost compared to FedPM and SignSGD, respectively (together with the accuracy boost over vanilla SignSGD). QSGD−KLMS, on the other hand, reduces the communication cost by 12 times over vanilla QSGD. Overall, among our baselines, QSGD requires the smallest bitrate, and FedPM achieves the highest accuracy. Surprisingly, FedPM−KLMS requires 10 times smaller bitrate than QSGD while achieving the same accuracy as FedPM at the same time – consistently in all the experiments. The consistent and significant improvements over DP−REC (in both bitrate and accuracy) justify the importance of (i) carefully choosing the pre-data and post-data distributions, and (ii) the adaptive block selection that optimizes the bit allocation.

4. Discussion & Conclusion

We introduced KLMS – a recipe for reducing the communication cost in stochastic FL frameworks by exploiting the side information available to the server and correlated with the local model updates. We highlighted the existence of highly natural choices of pre-data and post-data distribution in FL that we can take advantage of to reduce the communication cost significantly. Moreover, we showed how to adaptively adjust the bitrate across the model parameters and training rounds to achieve the fundamental communication cost – the

KL divergence between the pre-data and post-data distributions. While we showed four KLMS adaptations (that reduce the communication cost 50 times more than our baselines), it can be adapted to many other stochastic FL frameworks with similar communication gains.

References

Aji, A. and Heafield, K. Sparse communication for distributed gradient descent. In *EMNLP 2017: Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics (ACL), 2017.

Alistarh, D., Grubic, D., Li, J., Tomioka, R., and Vojnovic, M. QSGD: Communication-efficient SGD via gradient quantization and encoding. *Advances in Neural Information Processing Systems*, 2017.

Barnes, L. P., Inan, H. A., Isik, B., and Özgür, A. rtop-k: A statistical estimation approach to distributed SGD. *IEEE Journal on Selected Areas in Information Theory*, 1(3): 897–907, November 2020.

Basat, R. B., Vargaftik, S., Portnoy, A., Einziger, G., Ben-Itzhak, Y., and Mitzenmacher, M. QUICK-FL: Quick unbiased compression for federated learning. *arXiv preprint arXiv:2205.13341*, 2022.

Bernstein, J., Wang, Y.-X., Azizzadenesheli, K., and Anandkumar, A. signsgd: Compressed optimisation for non-convex problems. In *International Conference on Machine Learning*, pp. 560–569. PMLR, 2018.

Chatterjee, S. and Diaconis, P. The sample size required in importance sampling. *The Annals of Applied Probability*, 28(2):1099–1135, 2018.

Chen, H.-Y. and Chao, W.-L. Fed{be}: Making bayesian model ensemble applicable to federated learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=dgtpE6gKjHn.

Cohen, G., Afshar, S., Tapson, J., and Van Schaik, A. EMNIST: Extending MNIST to handwritten letters. In *International Joint Conference on Neural Networks (IJCNN)*, pp. 2921–2926, 2017.

Deng, L. The MNIST database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

El Mekkaoui, K., Mesquita, D., Blomstedt, P., and Kaski, S. Distributed stochastic gradient MCMC for federated learning. *arXiv preprint arXiv:2004.11231*, 2020.

El Mekkaoui, K., Parente Paiva Mesquita, D., Blomstedt, P., and Kask, S. Federated stochastic gradient langevin dynamics. In *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, pp. 1703–1712. PMLR, 2021.

Elias, P. Universal codeword sets and representations of the integers. *IEEE Transactions on Information Theory*, 21 (2):194–203, March 1975.

Flamich, G., Havasi, M., and Hernández-Lobato, J. M. Compressing images by encoding their latent representations with relative entropy coding. *Advances in Neural Information Processing Systems*, 33:16131–16141, 2020.

Flamich, G., Markou, S., and Hernández-Lobato, J. M. Fast relative entropy coding with a* coding. In *International Conference on Machine Learning*, pp. 6548–6577. PMLR, 2022.

Harsha, P., Jain, R., McAllester, D., and Radhakrishnan, J. The communication complexity of correlation. In *Twenty-Second Annual IEEE Conference on Computational Complexity (CCC'07)*, pp. 10–23. IEEE, 2007.

Havasi, M., Peharz, R., and Hernández-Lobato, J. M. Minimal random code learning: Getting bits back from compressed model parameters. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=r1f0YiCctm.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Isik, B., Weissman, T., and No, A. An information-theoretic justification for model pruning. In *International Conference on Artificial Intelligence and Statistics*, pp. 3821–3846. PMLR, 2022.

Isik, B., Pase, F., Gunduz, D., Koyejo, S., Weissman, T., and Zorzi, M. Communication-efficient federated learning through importance sampling. *arXiv preprint arXiv:2306.12625*, 2023a.

Isik, B., Pase, F., Gunduz, D., Weissman, T., and Zorzi, M. Sparse random networks for communication-efficient federated learning. In *The Eleventh International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=k1FHgri5y3-.

Jhunjhunwala, D., Mallick, A., Gadhikar, A., Kadhe, S., and Joshi, G. Leveraging spatial and temporal correlations in sparsified mean estimation. *Advances in Neural Information Processing Systems*, 34:14280–14292, 2021.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

Konečnỳ, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., and Bacon, D. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Li, A., Sun, J., Wang, B., Duan, L., Li, S., Chen, Y., and Li, H. Lotteryfl: Personalized and communication-efficient federated learning with lottery ticket hypothesis on non-iid datasets. *arXiv preprint arXiv:2008.03371*, 2020.

Li, A., Sun, J., Zeng, X., Zhang, M., Li, H., and Chen, Y. Fedmask: Joint computation and communication-efficient personalized federated learning via heterogeneous masking. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*, pp. 42–55, 2021.

Lin, Y., Han, S., Mao, H., Wang, Y., and Dally, B. Deep gradient compression: Reducing the communication bandwidth for distributed training. In *International Conference on Learning Representations*, 2018.

Liu, Y., Zhao, Y., Zhou, G., and Xu, K. Fedprune: Personalized and communication-efficient federated learning on non-iid data. In *International Conference on Neural Information Processing*, pp. 430–437. Springer, 2021.

Mitchell, N., Ballé, J., Charles, Z., and Konečnỳ, J. Optimizing the communication-accuracy trade-off in federated learning with rate-distortion theory. *arXiv preprint arXiv:2201.02664*, 2022.

Mohtashami, A., Jaggi, M., and Stich, S. Masked training of neural networks with partial gradients. In *International Conference on Artificial Intelligence and Statistics*, pp. 5876–5890. PMLR, 2022.

Mozaffari, H., Shejwalkar, V., and Houmansadr, A. FRL: Federated rank learning. *arXiv preprint arXiv:2110.04350*, 2021.

Ozfatura, E., Ozfatura, K., and Gündüz, D. Time-correlated sparsification for communication-efficient federated learning. In *IEEE International Symposium on Information Theory (ISIT)*, pp. 461–466. IEEE, 2021.

Plassier, V., Vono, M., Durmus, A., and Moulines, E. DG-LMC: a turn-key and scalable synchronous distributed MCMC algorithm via Langevin Monte Carlo within Gibbs. In *International Conference on Machine Learning*, pp. 8577–8587. PMLR, 2021.

Shah, A., Chen, W.-N., Balle, J., Kairouz, P., and Theis, L. Optimal compression of locally differentially private mechanisms. In *International Conference on Artificial Intelligence and Statistics*, pp. 7680–7723. PMLR, 2022.

Suresh, A. T., Felix, X. Y., Kumar, S., and McMahan, H. B. Distributed mean estimation with limited communication. In *International Conference on Machine Learning*, pp. 3329–3337. PMLR, 2017.

Theis, L. and Ahmed, N. Y. Algorithms for the communication of samples. In *International Conference on Machine Learning*, pp. 21308–21328. PMLR, 2022.

Triastcyn, A., Reisser, M., and Louizos, C. DP-REC: Private & communication-efficient federated learning. *arXiv preprint arXiv:2111.05454*, 2021.

Vallapuram, A. K., Zhou, P., Kwon, Y. D., Lee, L. H., Xu, H., and Hui, P. Hidenseek: Federated lottery ticket via server-side pruning and sign supermask. *arXiv preprint arXiv:2206.04385*, 2022.

Vargaftik, S., Ben-Basat, R., Portnoy, A., Mendelson, G., Ben-Itzhak, Y., and Mitzenmacher, M. Drive: one-bit distributed mean estimation. *Advances in Neural Information Processing Systems*, 34:362–377, 2021.

Vargaftik, S., Basat, R. B., Portnoy, A., Mendelson, G., Itzhak, Y. B., and Mitzenmacher, M. Eden: Communication-efficient and robust distributed mean estimation for federated learning. In *International Conference on Machine Learning*, pp. 21984–22014. PMLR, 2022.

Vogels, T., Karimireddy, S. P., and Jaggi, M. Powersgd: Practical low-rank gradient compression for distributed optimization. *Advances in Neural Information Processing Systems*, 32, 2019.

Vono, M., Plassier, V., Durmus, A., Dieuleveut, A., and Moulines, E. QLSD: Quantised Langevin stochastic dynamics for Bayesian federated learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 6459–6500. PMLR, 2022.

Wang, H., Sievert, S., Liu, S., Charles, Z., Papailiopoulos, D., and Wright, S. Atomo: Communication-efficient learning via atomic sparsification. *Advances in Neural Information Processing Systems*, 31, 2018.

Welling, M. and Teh, Y. W. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 681–688, 2011.

Wen, W., Xu, C., Yan, F., Wu, C., Wang, Y., Chen, Y., and Li, H. Terngrad: Ternary gradients to reduce communication in distributed deep learning. *Advances in Neural Information Processing Systems*, 30, 2017.

A. Related Work

A.1. Setups: Examples of Stochastic FL Frameworks

We now briefly summarize three examples of stochastic FL frameworks that KLMS can be integrated into by highlighting the natural choices for pre-data p_θ and post-data $q_{\phi^{(n)}}$ distributions.

A.1.1. FedPM (Isik et al., 2023b)

FedPM (Isik et al., 2023b) freezes the parameters of a randomly initialized network and finds a subnetwork inside it that performs well with the initial random parameters. To find the subnetwork, the clients receive a global probability mask $\theta^{(t)} \in [0,1]^d$ from the server that determines, for each parameter, the probability of retaining it in the subnetwork; set this as their local probability mask $\phi^{(t,n)} \leftarrow \theta^{(t)}$; and train only this mask (not the frozen random parameters) during local training. At inference, a sample $x^{(t,n)} \in \{0,1\}^d$ from the Bernoulli distribution $\text{Bern}(\cdot;\phi^{(t,n)})$ is taken, and multiplied element-wise with the frozen parameters of the network, obtaining a pruned random subnetwork, which is then used to compute the model outputs. Communication consists of three stages: (i) clients update their local probability masks $\phi^{(t,n)}$ through local training; (ii) at the end of local training, they send a sample $x^{(t,n)} \sim \text{Bern}(\cdot;\phi^{(t,n)})$ to the server; (iii) the server aggregates the samples $\frac{1}{N}\sum_{n=1}^{N} x^{(t,n)}$, updates the global probability mask $\theta^{(t+1)}$, and broadcasts the new mask to the clients for the next round. FedPM achieves state-of-the-art results in accuracy-bitrate tradeoff with around 1 bit per parameter (bpp). (We provide the pseudocode for FedPM in Algorithms 2 and 3. See (Isik et al., 2023b) for more details.) As the model converges, the global probability mask $\theta^{(t)}$ and clients' local probability masks $\phi^{(t,n)}$ get closer to each other (see Figures 2 and 5 for the trend of $D_{KL}(q_{\phi^{(t,n)}}||p_{\theta^{(t)}})$ over time). However, no matter how close they are, FedPM employs approximately the same bitrate for communicating a sample from $\text{Bern}(\cdot;\phi^{(t,n)})$ to the server that knows $p_{\theta^{(t)}}$. We show that this strategy is suboptimal and applying KLMS with the global probability distribution $\text{Bern}(\cdot;\theta^{(t)})$ as the pre-data distribution $p_{\theta^{(t)}}$, and the local probability distribution $\text{Bern}(\cdot;\phi^{(t,n)})$ as the post-data distribution $q_{\phi^{(t,n)}}$, provides up to 50 times gain in compression.

A.1.2. QSGD (Alistarh et al., 2017)

QSGD (Alistarh et al., 2017), different from the stochastic approach taken by FedPM to train a probabilistic mask, is proposed to train a deterministic set of parameters. However, QSGD is itself a stochastic quantization operation. More concretely, QSGD quantizes each coordinate $\mathbf{v}_i^{(t,n)}$ using the following probability distribution (which we call the QSGD distribution $p_{\text{QSGD}}(\cdot)$), where s is the number of quantization levels:

$$p_{\text{QSGD}}\left(\hat{\mathbf{v}}_i^{(t,n)}\right) = \begin{cases} \frac{s|\mathbf{v}_i^{(t,n)}|}{\|\mathbf{v}^{(t,n)}\|} - \left\lfloor \frac{s|\mathbf{v}_i^{(t,n)}|}{\|\mathbf{v}^{(t,n)}\|} \right\rfloor & \text{if } \hat{\mathbf{v}}_i^{(t,n)} = \frac{\|\mathbf{v}^{(t,n)}\|\cdot\text{sign}(\mathbf{v}_i^{(t,n)})}{s}\left(\left\lfloor \frac{s|\mathbf{v}_i^{(t,n)}|}{\|\mathbf{v}^{(t,n)}\|} \right\rfloor + 1\right) \\ 1 - \frac{s|\mathbf{v}_i^{(t,n)}|}{\|\mathbf{v}^{(t,n)}\|} + \left\lfloor \frac{s|\mathbf{v}_i^{(t,n)}|}{\|\mathbf{v}^{(t,n)}\|} \right\rfloor & \text{if } \hat{\mathbf{v}}_i^{(t,n)} = \frac{\|\mathbf{v}^{(t,n)}\|\cdot\text{sign}(\mathbf{v}_i^{(t,n)})}{s}\left\lfloor \frac{s|\mathbf{v}_i^{(t,n)}|}{\|\mathbf{v}^{(t,n)}\|} \right\rfloor \end{cases} . \tag{2}$$

(We provide the pseudocode for QSGD in Algorithm 4. See (Alistarh et al., 2017) for more details.) QSGD takes advantage of the empirical distribution of the quantized values (large quantized values are less frequent) by using Elias coding to encode them – which is the preferred code when the small values to encode are much more frequent than the larger values (Elias, 1975). However, QSGD still does not fully capture the distribution of the quantized values since Elias coding is not adaptive to the data. We fix this mismatch by applying KLMS with the QSGD distribution $p_{\text{QSGD}}(\cdot)$ as the post-data distribution $q_{\phi^{(t,n)}}$, and the empirical distribution induced by the historical updates at the server from the previous round as the pre-data distribution $p_{\theta^{(t)}}$. These two distributions are expected to be *close* to each other due to the temporal correlation across rounds, as previously reported by (Jhunjhunwala et al., 2021; Ozfatura et al., 2021). We demonstrate that KLMS exploits this closeness and outperforms vanilla QSGD with a 12 times improvement in bitrate.

A.1.3. Federated SGLD (Vono et al., 2022)

Federated SGLD (El Mekkaoui et al., 2021) targets a Bayesian FL setup, where the goal is to learn a global posterior distribution p_θ over the model parameters from clients' local posteriors $q_{\phi^{(n)}}$. A state-of-the-art method proposed in (Vono et al., 2022) is the federated counterpart of the Stochastic Gradient Langevin Dynamics (SGLD) (Welling & Teh, 2011), which uses a novel Markov Chain Monte Carlo (MCMC) algorithm. In this setting, the global posterior distribution is

Algorithm 2 Federated Probablistic Mask Training (FedPM) (Isik et al., 2023b).

Hyperparameters: local learning rate η_L, minibatch size B, number of local iterations τ.
Inputs: local datasets $\mathcal{D}_i$, $i = 1, \dots, N$, number of iterations T.
Output: random SEED and binary mask parameters $\boldsymbol{m}^{\text{final}}$.

At the server, initialize a random network with weight vector $\boldsymbol{w}^{\text{init}} \in \mathbb{R}^d$ using a random SEED, and broadcast it to the clients.
At the server, initialize the random score vector $\boldsymbol{s}^{(0,g)} \in \mathbb{R}^d$, and compute $\theta^{(0,g)} \leftarrow \text{Sigmoid}(\boldsymbol{s}^{(0,g)})$.
At the server, initialize Beta priors $\boldsymbol{\alpha}^{(0)} = \boldsymbol{\beta}^{(0)} = \boldsymbol{\lambda}_0$.
for $t = 1, \dots, T$ **do**
  Sample a subset $\mathcal{C}_t \subset \{1, \dots, N\}$ of $|\mathcal{C}_t| = C$ clients without replacement.
  On Client Nodes:
  for $c \in \mathcal{C}_t$ **do**
    Receive $\theta^{(t-1,g)}$ from the server and set $\boldsymbol{s}^{(t,c)} \leftarrow \text{Sigmoid}^{-1}(\theta^{(t-1,g)})$.
    for $l = 1, \dots, \tau$ **do**
      $\phi^{(t,c)} \leftarrow \text{Sigmoid}(\boldsymbol{s}^{(t,c)})$
      Sample binary mask $\boldsymbol{m}^{(t,c)} \sim q_{\boldsymbol{m}^{(t,c)}} = \text{Bern}(\phi^{(t,c)})$.
      $\dot{\boldsymbol{w}}^{(t,c)} \leftarrow \boldsymbol{m}^{(t,c)} \odot \boldsymbol{w}^{\text{init}}$
      $g_{\boldsymbol{s}^{(t,c)}} \leftarrow \frac{1}{B} \sum_{b=1}^{B} \nabla \ell(\dot{\boldsymbol{w}}^{(t,c)}; \mathcal{S}_b^c)$; where $\{\mathcal{S}_b^c\}_{b=1}^B$ are uniformly chosen from $\mathcal{D}_c$
      $\boldsymbol{s}^{(t,c)} \leftarrow \boldsymbol{s}^{(t,c)} - \eta_L \cdot g_{\boldsymbol{s}^{(t,c)}}$
    end for
    $\phi^{(t,c)} \leftarrow \text{Sigmoid}(\boldsymbol{s}^{(t,c)})$
    Sample a binary mask $\boldsymbol{m}^{(t,c)} \sim \text{Bern}(\phi^{(t,c)})$.
    Send the arithmetic coded binary mask $\boldsymbol{m}^{(t,c)}$ to the server.
  end for

  On the Server Node:
  Receive $\boldsymbol{m}^{(t,c)}$'s from C client nodes.
  $\theta^{(t,g)} \leftarrow \text{BayesAgg}(\{\boldsymbol{m}^{(t,c)}\}_{c \in \mathcal{C}_t}, t)$ // See Algorithm 3.
  Broadcast $\theta^{(t,g)}$ to all client nodes.
end for
Sample the final binary mask $\boldsymbol{m}^{\text{final}} \sim \text{Bern}(\theta^{(T,g)})$.
Generate the final model: $\dot{\boldsymbol{w}}^{\text{final}} \leftarrow \boldsymbol{m}^{\text{final}} \odot \boldsymbol{w}^{\text{init}}$.

Algorithm 3 BayesAgg. (Isik et al., 2023b)

Inputs: clients' updates $\{\boldsymbol{m}^{(t,c)}\}_{c \in \mathcal{C}_t}$, and round number t
Output: global probability mask $\boldsymbol{\pi}^{(t)}$

if ResPriors(t) **then**
  $\boldsymbol{\alpha}^{(t-1)} \leftarrow \boldsymbol{\beta}^{(t-1)} = \boldsymbol{\lambda}_0$
end if
Compute $\boldsymbol{m}^{(t,\text{agg})} = \sum_{k \in \mathcal{C}_t} \boldsymbol{m}^{(t,c)}$.
$\boldsymbol{\alpha}^{(t)} \leftarrow \boldsymbol{\alpha}^{(t-1)} + \boldsymbol{m}^{(t,\text{agg})}$
$\boldsymbol{\beta}^{(t)} \leftarrow \boldsymbol{\beta}^{(t-1)} + C \cdot \mathbf{1} - \boldsymbol{m}^{(t,\text{agg})}$
$\boldsymbol{\pi}^{(t)} \leftarrow \frac{\boldsymbol{\alpha}^{(t-1)}}{\boldsymbol{\alpha}^{(t)} + \boldsymbol{\beta}^{(t)} - 2}$
Return $\boldsymbol{\pi}^{(t)}$

assumed to be proportional to the product $p_{\theta^{(t)}} \sim \prod_{n=1}^{N} e^{-U(\phi^{(t,n)})}$ of N local unnormalized posteriors associated with each client, expressed as potential functions $\{U(\phi^{(t,n)})\}_{n=1}^N$. At the beginning of each local training round, the local clients' posteriors are initialized with the global posterior $\phi^{(n,t)} \leftarrow \theta^{(t)}$, $\forall n \in [N]$. Then the clients compute an unbiased estimate of their gradients $H(\phi^{(t,n)}) = \frac{|D^{(n)}|}{|\mathcal{S}^{(t,n)}|} \sum_{j \in \mathcal{S}^{(t,n)}} \nabla U_j(\phi^{(t,n)})$, where $|D^{(n)}|$ is the size of the local dataset of client n, and

Algorithm 4 Quantized Stochastic Gradient Descent (QSGD) (Alistarh et al., 2017).

Hyperparameters: server learning rate η_S, local learning rate η_S, number of quantization levels s, minibatch size B.
Inputs: local datasets $\mathcal{D}_n$, $n = 1, \dots, N$, number of iterations T.
Output: final model $\boldsymbol{w}^{(T)}$.

At the server, initialize a random network with weight vector $\boldsymbol{w}^{(0,g)} \in \mathbb{R}^d$ and broadcast it to the clients.
for $t = 1, \dots, T$ **do**
  Sample a subset $\mathcal{C}_t \subset \{1, \dots, N\}$ of $|\mathcal{C}_t| = C$ clients without replacement.
  On Client Nodes:
  for $c \in \mathcal{C}_t$ **do**
    Receive $\boldsymbol{w}^{(t-1,g)}$ from the server and set the local model parameters $\boldsymbol{w}^{(t,c)} \leftarrow \boldsymbol{w}^{(t,g)}$.
    for $l = 1, \dots, \tau$ **do**
      $g_w^{(t,c)} \leftarrow \frac{1}{B} \sum_{b=1}^{B} \nabla \ell(\boldsymbol{w}^{(t,c)}; \mathcal{S}_b^c)$; where $\{\mathcal{S}_b^c\}_{b=1}^B$ are uniformly chosen from $\mathcal{D}_c$
      $\boldsymbol{w}^{(t,c)} \leftarrow \boldsymbol{w}^{(t,c)} - \eta_L \cdot g_w^{(t,c)}$
    end for
    $\mathbf{v}^{(t,c)} \leftarrow \boldsymbol{w}^{(t,c)} - \boldsymbol{w}^{(t,g)}$
    for $i = 1, \dots, d$ **do**
      Find integer $0 \leq q \leq s$ such that $|\mathbf{v}_i^{(t,c)}|/\|\mathbf{v}^{(t,c)}\|_2 \in [q/s, (q+1)/s]$.
      Take a sample $z \sim \text{Bern}(1 - (\frac{|\mathbf{v}_i^{(t,c)}|}{\|\mathbf{v}^{(t,c)}\|_2} s - q))$.
      if $z = 1$ **then**
        $\kappa_i^{(t,c)} \leftarrow q/s$.
      else
        $\kappa_i^{(t,c)} \leftarrow (q+1)/s$.
      end if
    end for
    Send vectors $\kappa^{(t,c)}$, $\text{sign}(\mathbf{v}^{(t,c)})$, and norm $\|\mathbf{v}^{(t,n)}\|_2$ to the server using Elias coding (Elias, 1975) as in (Alistarh et al., 2017).
  end for

  On the Server Node:
  Receive $\kappa^{(t,c)}$, $\text{sign}(\mathbf{v}^{(t,c)})$, and norm $\|\mathbf{v}^{(t,c)}\|_2$ from the clients $c \in \mathcal{C}_t$.
  for $c \in \mathcal{C}_t$ **do**
    for $i = 1, \dots, d$ **do**
      Reconstruct $\hat{\mathbf{v}}_i^{(t,c)} \leftarrow \|\mathbf{v}^{(t,c)}\|_2 \cdot \text{sign}(\mathbf{v}_i^{(t,c)}) \cdot \kappa_i^{(t,c)}$.
    end for
  end for
  Aggregate and update $\boldsymbol{w}^{(t,g)} \leftarrow \boldsymbol{w}^{(t-1,g)} - \eta_S \frac{1}{C} \sum_{c \in \mathcal{C}_t} \hat{\mathbf{v}}^{(t,c)}$.
  Broadcast $\boldsymbol{w}^{(t,g)}$ to the clients.
end for

$\mathcal{S}^{(t,n)}$ is the batch of data used to estimate the gradient. They then communicate these estimates to the server, which aggregates them by computing

$$\theta^{(t+1)} = \theta^{(t)} - \gamma \sum_{n=1}^{N} H(\phi^{(t,n)}) + \sqrt{2\gamma}\xi^{(t)}, \tag{3}$$

where $\xi^{(t)}$ is a sequence of i.i.d. standard Gaussian random variables. As reported in (El Mekkaoui et al., 2021; Vono et al., 2022), the sequence of global updates $\theta^{(t)}$ converges to the posterior sampling. Notice that the clients communicate their gradient vectors $H(\phi^{(t,n)})$ to the server at every round, which is as large as the model itself. To reduce this communication cost, in (Vono et al., 2022), the authors propose a compression algorithm called QLSD that stochastically quantizes the updates with essentially the Bayesian counterpart of QSGD (Alistarh et al., 2017). (We provide the pseudocode for QLSD in Algorithm 5. See (Vono et al., 2022) for more details.) However, neither QLSD nor the other compression baselines in the Bayesian FL literature (Chen & Chao, 2021; El Mekkaoui et al., 2020; Plassier et al., 2021) take full advantage of

the stochastic formulation of the Bayesian framework, where the server and the clients share side information (the global posterior $p_{\theta^{(t)}}$) that could be used to improve the compression gains. Instead, they quantize the updates ignoring this side information. This approach is suboptimal since (i) the precision is already degraded in the quantization step, and (ii) the compression step does not account for the side information $p_{\theta^{(t)}}$. We show that we can exploit this inherent stochastic formulation of Bayesian FL by applying KLMS with the global posterior distribution as the pre-data distribution $p_{\theta^{(t)}}$, and the local posterior distribution as the post-data distribution $q_{\phi^{(t,n)}}$. In addition to benefiting from the side information, KLMS does not restrict the message domain to be discrete (as opposed to the baselines) and can reduce the communication cost by 4 times, while also achieving higher accuracy than the baselines.

Algorithm 5 Quantised Langevin Stochastic Dynamics (QLSD) (Vono et al., 2022).

Hyperparameters: server learning rate η_S, number of quantization levels s, minibatch size B.
Inputs: local datasets $\mathcal{D}_n$, $n = 1, \dots, N$, number of iterations T.
Output: samples $\left\{\theta^{(t)}\right\}_{t=1}^{T}$.

At the server, initialize a random network with weight vector $\theta^{(0)} \in \mathbb{R}^d$ and broadcast it to the clients.
for $t = 1, \dots, T$ **do**
  Sample a subset $\mathcal{C}_t \subset \{1, \dots, N\}$ of $|\mathcal{C}_t| = C$ clients without replacement.
  On Client Nodes:
  for $c \in \mathcal{C}_t$ **do**
    Receive $\theta^{(t-1)}$ from the server and set the local model parameters $\phi^{(t,c)} \leftarrow \theta^{(t-1)}$.
    Sample a minibatch $\mathcal{S}^c$ s.t. $|\mathcal{S}^c| = B$ uniformly from $\mathcal{D}_c$.
    Compute a stochastic gradient of the potential $H(\phi^{(t,c)}) \leftarrow \frac{|D_j^{(c)}|}{B} \sum_{j \in \mathcal{S}^c} \nabla U_j(\phi^{(t,c)})$.
    for $i = 1, \dots, d$ **do**
      Find integer $0 \leq q \leq s$ such that $\frac{|H_i(\phi^{(t,c)})|}{\|H(\phi^{(t,c)})\|_2} \in [q/s, (q+1)/s]$.
      Take a sample $z \sim \text{Bern}(1 - (\frac{|H_i(\phi^{(t,c)})|}{\|H(\phi^{(t,c)})\|_2} s - q))$.
      if $z = 1$ **then**
        $\kappa_i^{(t,c)} \leftarrow q/s$.
      else
        $\kappa_i^{(t,c)} \leftarrow (q+1)/s$.
      end if
    end for
    Send vectors $\kappa^{(t,c)}$, $\text{sign}(H(\phi^{(t,c)}))$, and norm $\|H(\phi^{(t,c)})\|_2$ to the server using Elias coding (Elias, 1975) as in (Alistarh et al., 2017).
  end for

  On the Server Node:
  Receive $\kappa^{(t,c)}$, $\text{sign}(H(\phi^{(t,c)}))$, and norm $\|H(\phi^{(t,c)})\|_2$ from the clients $c \in \mathcal{C}_t$.
  for $c \in \mathcal{C}_t$ **do**
    for $i = 1, \dots, d$ **do**
      Reconstruct $\hat{H}_i(\phi^{(t,c)}) \leftarrow \|H(\phi^{(t,c)})\|_2 \cdot \text{sign}(H_i(\phi^{(t,c)})) \cdot \kappa_i^{(t,c)}$.
    end for
  end for
  Compute $\hat{H}(\phi^{(t)}) \leftarrow \frac{N}{C} \sum_{c \in \mathcal{C}_t} \hat{H}(\phi^{(t,c)})$.
  Sample $\xi^{(t)} \sim \mathcal{N}(\mathbf{0}_d, \boldsymbol{I}_d)$.
  Compute $\theta^{(t)} \leftarrow \theta^{(t-1)} - \eta_S \hat{H}(\phi^{(t)}) + \sqrt{2\gamma}\xi^{(t)}$.
  Broadcast $\theta^{(t)}$ to the clients.
end for

A.2. Other Related Work

In this section, we briefly discuss the related work in (i) communication-efficient FL and (ii) importance sampling literatures.

Communication-Efficient FL: There has been extensive research in reducing the communication cost of FL (i) by compressing the model updates through sparsification (Aji & Heafield, 2017; Barnes et al., 2020; Lin et al., 2018; Ozfatura et al., 2021; Wang et al., 2018), quantization (Mitchell et al., 2022; Suresh et al., 2017; Vono et al., 2022; Wen et al., 2017), and low-rank factorization (Basat et al., 2022; Mohtashami et al., 2022; Vogels et al., 2019; Wang et al., 2018); or (ii) by training sparse subnetworks instead of the full model (Isik et al., 2023b; Li et al., 2020; 2021; Liu et al., 2021; Mozaffari et al., 2021; Vallapuram et al., 2022). Among these approaches, those based on stochastic updates have shown success over the deterministic ones in similar settings. For instance, for finding sparse subnetworks within a large random model, `FedPM` (Isik et al., 2023b) takes a stochastic approach by training a probability mask, indicating the probability for each random model parameter to be part of the subnetwork, and extracting those subnetworks by taking samples from the distribution parameterized by the trained probability mask. With this approach, `FedPM` outperforms other methods that find sparse subnetworks (Li et al., 2021; Mozaffari et al., 2021; Vallapuram et al., 2022) with significant accuracy and bitrate gains. Similarly, for the standard FL setting (training model parameters), `QSGD` (Alistarh et al., 2017) is an effective stochastic quantization method – outperforming most other quantization schemes such as `SignSGD` (Bernstein et al., 2018) and `TernGrad` (Wen et al., 2017) by large margins. Lastly, in the Bayesian FL setting, `QLSD` (Vono et al., 2022) proposes a Bayesian counterpart of `QSGD`, and performs better than other baselines (Chen & Chao, 2021; El Mekkaoui et al., 2020; Plassier et al., 2021). While all these stochastic approaches already perform better than the relevant baselines, in this work, we show that they do not take full advantage of the *side information* available to the server. We provide a guideline on how to find useful side information under each setting and introduce `KLMS` that reduces the communication cost to the fundamental distance between the client's distribution that they want to communicate samples from and the side information at the server (with 50 times reduced communication cost compared to the baselines).

Importance Sampling: Our strategy is inspired by the importance sampling algorithm studied in (Chatterjee & Diaconis, 2018; Harsha et al., 2007; Theis & Ahmed, 2022), and later applied for model compression (Havasi et al., 2019), learned image compression (Flamich et al., 2020; 2022), and compressing differentially private mechanisms (Shah et al., 2022; Triastcyn et al., 2021). One relevant work to ours is (Havasi et al., 2019), which applies the importance sampling strategy in (Chatterjee & Diaconis, 2018) to compress Bayesian neural networks. Since the model size is too large to be compressed at once, they compress fixed-size blocks of the model parameters separately and independently. As we elaborate in Section 2, this can be done much more efficiently by choosing the block size adaptively based on the information content of each parameter. While this adaptive strategy could bring some extra communication overhead when applied for model compression (to locate the adaptive-size blocks), we explain how to avoid this overhead in the FL setting by exploiting temporal correlations. Another relevant work is `DP-REC` (Triastcyn et al., 2021), which again applies the importance sampling technique in (Chatterjee & Diaconis, 2018) to compress the model updates in FL, while also showing differential privacy implications. However, since their training strategy is fully deterministic (no probabilistic learning or stochastic compression), the choice of pre-data and post-data distributions is somewhat arbitrary. Instead, in our work, the goal is to exploit the available side information to the full extent by choosing natural pre-data and post-data distributions – which improves the communication efficiency over `DP-REC` significantly. Another factor in this improvement is the adaptive bit allocation strategy mentioned above – which could actually be integrated into `DP-REC` as well by avoiding the extra communication overhead as we do in our work (since `DP-REC` works in an FL setting too). Our experimental results demonstrate that these two improvements are indeed critical for boosting the accuracy-bitrate tradeoff. Finally, we extend the theoretical guarantees of importance sampling, which quantifies the required bitrate for a target discrepancy (due to compression), to the distributed setting, where we can recover the existing results in (Chatterjee & Diaconis, 2018) as a special case by setting $N = 1$.

B. KLMS Pseudocode

In this section, we provide pseudocodes for both versions of KLMS: Algorithm 6 with fixed-sized blocks (Fixed-KLMS), and Algorithm 7 with adaptive-sized blocks (Adaptive-KLMS). The algorithms are standalone coding modules that can be applied to the different FL frameworks (see Appendix D). In the experiments in Section 3, we used Adaptive-KLMS and called it KLMS for simplicity. The decoding approach at the server is outlined in Algorithm 9.

Algorithm 6 Fixed-KLMS.

Inputs: post-data $q_{\phi^{(t,c)}}$ and pre-data $p_{\theta^{(t)}}$ distributions, block size S, number of per-block samples K.

Output: selected indices for each block $\{k_{[m]}^{(c)*}\}_{m=1}^{M}$, where $M = \lceil \frac{d}{S} \rceil$ is the number of bloks.

Define $\{q_{\phi_{[m]}^{(t,c)}}\}_{m=1}^{M}$ and $\{p_{\theta_{[m]}^{(t,c)}}\}_{m=1}^{M}$ splitting $q_{\phi^{(t,c)}}$ and $p_{\theta^{(t)}}$ into M distributions on S-size parameters blocks.
for all $m \in \{1, \dots, M\}$ **do**
  $I \leftarrow [(m-1)S : mS]$.
  Take K samples from the pre-data distribution: $\{\mathbf{y}_{[k]}\}_{k=1}^{K} \sim p_{\theta_{[I]}^{(t)}}$.
  $\alpha_{[k]} \leftarrow \frac{q_{\phi_{[I]}^{(t,c)}}(\mathbf{y}_{[k]})}{p_{\theta_{[I]}^{(t)}}(\mathbf{y}_{[k]})} \; \forall k \in \{1, \dots, K\}$.
  $\pi(k) \leftarrow \frac{\alpha_{[k]}}{\sum_{k'=1}^{K} \alpha_{[k']}} \; \forall k \in \{1, \dots, K\}$.
  Sample an index $k_{[m]}^{(c)*} \sim \pi(k)$.
end for
Send the selected indices $\{k_{[m]}^{(c)*}\}_{m=1}^{M}$ with $M \cdot \log_2 K$ bits overall for M blocks.

Algorithm 7 `Adaptive-KLMS`.

Inputs: post-data $q_{\phi^{(t,c)}}$ and pre-data $p_{\theta^{(t)}}$ distributions, block locations M (a list of start indices of each block), number of per-block samples K, target KL divergence D_{KL}^{target}, the flag `UPDATE` indicating whether the block locations will be updated, the maximum block size allowed `MAX_BLOCK_SIZE`.
Output: selected indices for each block $\{k_{[m]}^{(c)*}\}_{m=1}^{M}$, where the number of blocks M may vary each round.

if `UPDATE` **then**
  Construct the sequence of per-coordinate KL-divergence of size d: $\mathbf{D} \leftarrow \left[D_{KL}(q_{\phi_1^{(t,c)}} \| p_{\theta_1^{(t)}}), D_{KL}(q_{\phi_2^{(t,c)}} \| p_{\theta_2^{(t)}}), \ldots, D_{KL}(q_{\phi_d^{(t,c)}} \| p_{\theta_d^{(t)}})\right]$.
  Divide $\mathbf{D}$ into subsequences of $\{\mathbf{D}[i_1 = 1 : i_2], \mathbf{D}[i_2 : i_3], \ldots, \mathbf{D}[i_M : i_{M+1} = d]\}$ such that for all $m = 1, \ldots, M$, $\sum_{l=i_m}^{i_{m+1}} \mathbf{D}[l] \approx D_{KL}^{\text{target}}$ or $i_{m+1} - i_m = \texttt{MAX\_BLOCK\_SIZE}$. Here M, i.e, the number of blocks, may vary each round.
  Construct new block locations: $I_m \leftarrow [i_m : i_{m+1}]$ for $m = 1, \ldots, M$.
else
  Keep the old block locations I.
end if
Construct per-block post-data $\{q_{\phi_{[I_m]}^{(t,c)}}\}_{m=1}^{M}$ and pre-data $\{p_{\theta_{[I_m]}^{(t)}}\}_{m=1}^{M}$ distributions.
for all $m \in \{1, \ldots, M\}$ **do**
  Sample $\{\mathbf{y}_{[k]}\}_{k=1}^{K} \sim p_{\theta_{[I_m]}^{(t)}}$.
  $\alpha_{[k]} \leftarrow \frac{q_{\phi_{[I_m]}^{(t,c)}}(\mathbf{y}_{[k]})}{p_{\theta_{[I_m]}^{(t)}}(\mathbf{y}_{[k]})} \cdot \forall k \in \{1, \ldots, K\}$.
  $\pi(k) \leftarrow \frac{\alpha_{[k]}}{\sum_{k'=1}^{K} \alpha_{[k']}} \; \forall k \in \{1, \ldots, K\}$.
  Sample $k_{[m]}^{(c)*} \sim \pi(k)$.
end for
if `UPDATE` **then**
  Return the selected indices $\{k_{[m]}^{(c)*}\}_{m=1}^{M}$ and the new block locations I spending $\approx D_{KL}^{\text{target}} + \log_2(\texttt{MAX\_BLOCK\_SIZE})$ bits per block (block sizes are different for each block).
else
  Return the selected indices $\{k_{[m]}^{(c)*}\}_{m=1}^{M}$ spending $\approx D_{KL}^{\text{target}}$ bits per block (block sizes are different for each block).
end if

Algorithm 8 `Aggregate-Block-Locations`.

Inputs: client block locations $\{I^{(t,c)}\}_{c \in \mathcal{C}_t}$.
Output: new global block locations $I^{(t)}$.

Define empty $I^{(t)}$.
$m_{\max} \leftarrow \max_{c \in \mathcal{C}_t} \{\text{length}(I^{(t,c)})\}$.
for $m \in \{1, 2, \ldots, m_{\max}\}$ **do**
  $\tilde{i}_m \leftarrow 0$.
  $l \leftarrow 0$.
  for $c \in \mathcal{C}_t$ **do**
    if $\text{length}(I^{(t,c)}) \geq m$ **then**
      $\tilde{i}_m \leftarrow \tilde{i}_m + I_{i_m}^{(t,c)}$.
      $l \leftarrow l + 1$.
    end if
  end for
  $\bar{i}_m \leftarrow \lceil \tilde{i}_m / l \rceil$.
  Add $\bar{i}_m$ to $I^{(t)}$.
end for
Return $I^{(t)}$.

Algorithm 9 KLMS-Decoder.

Inputs: pre-data $p_{\theta^{(t)}}$ distribution, block locations I of M blocks, number of per-block samples K, selected indices for each block $\{k^{(c)*}_{[m]}\}_{m=1}^{M}$, where $M = \lceil \frac{d}{S} \rceil$ is the number of blocks.

Output: The selected samples $\{\mathbf{y}^{*}_{[m]}\}_{m=1}^{M}$ for each block.

Define $\{p_{\theta^{(t)}_{[I_m]}}\}_{m=1}^{M}$ splitting $p_{\theta^{(t)}}$ into M distributions with block locations in I.

for all $m \in \{1, \ldots, M\}$ **do**

Take K samples from the pre-data distribution: $\{\mathbf{y}_{[k]}\}_{k=1}^{K} \sim p_{\theta^{(t)}_{[I_m]}}$.

Recover $\mathbf{y}^{*}_{[m]} \leftarrow \mathbf{y}_{k^{(c)}_{[m]}}$.

end for

Return the selected samples $\{\mathbf{y}^{*}_{[m]}\}_{m=1}^{M}$ for each block.

C. Proofs

In this section, we provide the proof for Theorem 2.1. But before that, we first define the formal problem statement, introduce some new notation, and give another theorem (Theorem C.1) that will be required for the proof of Theorem 2.1.

We consider a scenario where N distributed nodes and a centralized server share a prior distribution p_θ over a set $\mathcal{X}$ equipped with some sigma algebra. Each node n also holds a posterior distribution $q_{\phi^{(n)}}$ over the same set. The server wants to estimate $\mathbb{E}_{X^{(n)}\sim q_{\phi^{(n)}}\forall n\in[N]}[\frac{1}{N}\sum_{m=1}^{N} f(X^{(m)})]$, where $f(\cdot):\mathcal{X}\to\mathbb{R}$ is a measurable function. In order to minimize the cost of communication from the nodes to the centralized server, each node n and the centralized server take $K^{(n)}$ samples from the prior distribution $\mathbf{y}_{[1]}^{(n)},\dots,\mathbf{y}_{[K^{(n)}]}^{(n)}\sim p_\theta$. Then client n performs the following steps:

1. Define a new probability distribution over the indices $k=1,\dots,K^{(n)}$:

$$\pi^{(n)}(k)=\frac{q_{\phi^{(n)}}(\mathbf{y}_{[k]}^{(n)})/p_\theta(\mathbf{y}_{[k]}^{(n)})}{\sum_{l=1}^{K^{(n)}} q_{\phi^{(n)}}(\mathbf{y}_{[l]}^{(n)})/p_\theta(\mathbf{y}_{[l]}^{(n)})} \tag{4}$$

   and over the samples $\mathbf{y}_{[1]}^{(n)},\dots,\mathbf{y}_{[K^{(n)}]}^{(n)}$:

$$\tilde{q}_{\pi^{(n)}}(\mathbf{y})=\sum_{k=1}^{K^{(n)}}\pi^{(n)}(k)\cdot\mathbf{1}\{\mathbf{y}_{[k]}^{(n)}=\mathbf{y}\}. \tag{5}$$

2. Sample $k^{(n)*}\sim\pi^{(n)}$.
3. Communicate $k^{(n)*}$ to the centralized server with $\log K^{(n)}$ bits.

Then, the centralized server recovers the sample $\mathbf{y}_{[k^{(n)*}]}^{(n)}$ that it generated in the beginning. (Note that $\mathbf{y}_{[k^{(n)*}]}^{(n)}$ is actually a sample from $\tilde{q}_{\pi^{(n)}}$.) Finally, the server aggregates these samples $\frac{1}{N}\sum_{n=1}^{N} f(\mathbf{y}_{k^{(n)*}}^{(n)})$ which is an estimate of

$$\mathbb{E}_{Y^{(n)}\sim\tilde{q}_{\pi^{(n)}}\forall n\in[N]}[\frac{1}{N}\sum_{m=1}^{N} f(Y^{(m)})]. \tag{6}$$

We want to find a relation between the number of samples $K^{(1)},\dots,K^{(N)}$ (or the number of bits $\log K^{(1)},\dots,\log K^{(N)}$) and the error in the estimate, $|\mathbb{E}_{Y^{(n)}\sim\tilde{q}_{\pi^{(n)}}\forall n\in[N]}[\frac{1}{N}\sum_{m=1}^{N} f(Y^{(m)})]-\mathbb{E}_{X^{(n)}\sim q_{\phi^{(n)}}\forall n\in[N]}[\frac{1}{N}\sum_{m=1}^{N} f(X^{(m)})]|$. In our proofs, we closely follow the methodology in Theorems 1.1. and 1.2. in (Chatterjee & Diaconis, 2018). In Theorem C.1, we use the probability density of $q_{\phi^{(n)}}$ with respect to p_θ for each node n and denote it by $\rho_n=\frac{dq_{\phi^{(n)}}}{dp_\theta}$. We refer to the following definitions often:

$$I(f)=\int_{\mathbf{x}^{(1)}}\cdots\int_{\mathbf{x}^{(N)}}\left(\frac{1}{N}\sum_{n=1}^{N} f(\mathbf{x}^{(n)})\right)\prod_{n=1}^{N} dq_{\phi^{(n)}}(\mathbf{x}^{(n)}), \tag{7}$$

$$I_K(f)=\frac{1}{\prod_{n=1}^{N}K^{(n)}}\sum_{k^{(1)}=1}^{K^{(1)}}\cdots\sum_{k^{(N)}=1}^{K^{(N)}}\left(\frac{1}{N}\sum_{n=1}^{N} f(\mathbf{y}_{[k^{(n)}]}^{(n)})\right)\prod_{n=1}^{N}\rho_n(\mathbf{y}_{[k^{(n)}]}^{(n)}), \tag{8}$$

and

$$J_K(f) = \sum_{k^{(1)}=1}^{K^{(1)}} \cdots \sum_{k^{(N)}=1}^{K^{(N)}} \left(\frac{1}{N} \sum_{n=1}^{N} f(\mathbf{y}_{[k^{(n)}]}^{(n)}) \right) \prod_{n=1}^{N} \frac{q_{\phi^{(n)}}(\mathbf{y}_{[k^{(n)}]}^{(n)})/p_\theta(\mathbf{y}_{[k^{(n)}]}^{(n)})}{\sum_{l=1}^{K^{(n)}} q_{\phi^{(n)}}(\mathbf{y}_{[l]}^{(n)})/p_\theta(\mathbf{y}_{[l]}^{(n)})}. \tag{9}$$

Notice that $I(f)$ corresponds to the target value the centralized server wants to estimate, $J_K(f)$ is the estimate from the proposed approach, and $I_K(f)$ is a value that will be useful in the proof and that satisfies $\mathbb{E}[I_K(f)] = I(f)$.

Theorem C.1. *Let p_θ and $q_{\phi^{(n)}}$ for $n = 1, \ldots, N$ be probability distributions over a set $\mathcal{X}$ equipped with some sigma-algebra. Let $X^{(n)}$ be an $\mathcal{X}$-valued random variable with law $q_{\phi^{(n)}}$. Let $r \geq 0$ and $\tilde{q}_{\pi^{(n)}}$ for $n = 1, \ldots, N$ be discrete distributions each constructed by $K^{(n)} = \exp\left(D_{KL}(q_{\phi^{(n)}} \| p_\theta) + r\right)$ samples $\{\mathbf{y}_{[k^{(n)}]}^{(n)}\}_{k^{(n)}=1}^{K^{(n)}}$ from p_θ defining $\tilde{q}_{\pi^{(n)}}(\mathbf{y}) = \sum_{k=1}^{K^{(n)}} \frac{q_{\phi^{(n)}}(\mathbf{y}_{[k]}^{(n)})/p_\theta(\mathbf{y}_{[k]}^{(n)})}{\sum_{l=1}^{K^{(n)}} q_{\phi^{(n)}}(\mathbf{y}_{[l]}^{(n)})/p_\theta(\mathbf{y}_{[l]}^{(n)})} \cdot \mathbf{1}\{\mathbf{y}_{[k]}^{(n)} = \mathbf{y}\}$. Furthermore, for $f(\cdot)$ defined above, let $||f||_{\mathbf{q}_\phi} = \sqrt{\mathbb{E}_{X^{(n)} \sim q_{\phi^{(n)}} \forall n \in [N]}[(\frac{1}{N}\sum_{m=1}^{N} f(X^{(m)}))^2]}$ be its 2-norm under $\mathbf{q}_\phi = q_{\phi^{(1)}}, \ldots, q_{\phi^{(N)}}$. Then,*

$$\mathbb{E}|I_K(f) - I(f)| \leq ||f||_{\mathbf{q}_\phi} \left(e^{-Nr/4} + 2\sqrt{\prod_{n=1}^{N} \mathbb{P}\left(\log \rho_n(X^{(n)}) > D_{KL}(q_{\phi^{(n)}}||p_\theta) + r/2\right)} \right). \tag{10}$$

Conversely, let $\mathbf{1}$ denote the function from $\mathcal{X}$ into $\mathbb{R}$ that is identically equal to 1. If for $n = 1, \ldots, N$, $K^{(n)} = \exp\left(D_{KL}(q_{\phi^{(n)}}||p_\theta) - r\right)$ for some $r \geq 0$, then for any $\delta \in (0, 1)$,

$$\mathbb{P}(I_K(\mathbf{1}) \geq 1 - \delta) \leq e^{-Nr/2} + \frac{\prod_{n=1}^{N} \mathbb{P}\left(\log \rho_n(X^{(n)}) \leq D_{KL}(q_{\phi^{(n)}}||p_\theta) - r/2\right)}{1 - \delta}. \tag{11}$$

Proof. Let $L^{(n)} = D_{KL}(q_{\phi^{(n)}}||p_\theta), \forall n \in [N]$. Suppose that $K^{(n)} = e^{L^{(n)}+r}$ and $a^{(n)} = e^{L^{(n)}+r/2}$. Let $h(z) = f(z)$ if $\rho_n(z) \leq a^{(n)}$ and 0 otherwise $\forall n \in [N]$. We first make the following assumption:

$$\begin{aligned} \mathbb{E}[|\frac{1}{N} \sum_{n \in Q \subseteq [N]} f(X^{(n)})|; \forall n \in Q \subseteq [N], \rho_n(X^{(n)}) > a^{(n)}] \leq \\ \mathbb{E}[|\frac{1}{N} \sum_{n \in [N]} f(X^{(n)})|; \forall n \in [N], \rho_n(X^{(n)}) > a^{(n)}]. \end{aligned} \tag{12}$$

This is indeed a reasonable assumption. To see this, following (Chatterjee & Diaconis, 2018), we note that $\log \rho_n(Z)$ is concentrated around its expected value, which is $L^{(n)} = D_{KL}(q_{\phi^{(n)}}||p_\theta)$, in many scenarios. Therefore, for small t (and t is indeed negligibly small in our experiments), the events $\mathbf{1}\{\forall n \in Q \subseteq [N], \rho_n(X^{(n)}) > a^{(n)}\}$ occur with the approximately same frequency for each set $Q \subseteq [N]$ since the likelihood of event $\mathbf{1}\{\rho_n(X^{(n)}) > a^{(n)}\}$ is close to being uniform. Consider also that $|\frac{1}{N}\sum_{n \in Q \subseteq [N]} f(X^{(n)})| \leq |\frac{1}{N}\sum_{n \in [N]} f(X^{(n)})|$ holds when $f(X^n)$'s have the same signs per coordinate for each $n = 1, \ldots, N$, which is a realistic assumption given that the clients are assumed to be able to train a joint model and hence should not have opposite signs in the updates very often. With these two observations, we argue that the assumption in (12) is indeed reasonable for many scenarios, including FL.

Now, going back to the proof, from triangle inequality, we have,

$$|I_K(f) - I(f)| \leq |I_K(f) - I_K(h)| + |I_K(h) - I(h)| + |I(h) - I(f)|. \tag{13}$$

First, note that by Cauchy-Schwarz inequality and by the assumption in (12), we have

$$|I(h)-I(f)| = \sum_{Q\subseteq[N]} \mathbb{E}[|\frac{1}{N}\sum_{m\in Q} f(X^{(m)})|; \forall n\in Q, \rho_n(X^{(n)}) > a^{(n)}]\cdot \cdot \mathbb{P}(\forall n\in Q, \rho_n(X^{(n)}) > a^{(n)}) \tag{14}$$

$$\leq \mathbb{E}[|\frac{1}{N}\sum_{m\in[N]} f(X^{(m)})|; \forall n\in[N], \rho_n(X^{(n)}) > a^{(n)}] \sum_{Q\subseteq[N]} \mathbb{P}(\forall n\in Q, \rho_n(X^{(n)}) > a^{(n)}) \tag{15}$$

$$= \mathbb{E}[|\frac{1}{N}\sum_{m\in[N]} f(X^{(m)})|; \forall n\in[N], \rho_n(X^{(n)}) > a^{(n)}] \tag{16}$$

$$= \int_{\mathbf{x}^{(1)},\dots,\mathbf{x}^{(N)}} |\frac{1}{N}\sum_{n=1}^{N} f(\mathbf{x}^{(n)})| \cdot 1\{\forall n\in[N], \rho_n(\mathbf{x}^{(n)}) > a^{(n)}\} \prod_{n=1}^{N} dq_{\phi^{(n)}}(\mathbf{x}^{(n)}) \tag{17}$$

$$\leq \sqrt{\int_{\mathbf{x}^{(1)},\dots,\mathbf{x}^{(N)}} |\frac{1}{N}\sum_{m=1}^{N} f(\mathbf{x}^{(m)})|^2 \cdot \prod_{n=1}^{N} dq_{\phi^{(n)}}(\mathbf{x}^{(n)})} \cdot \sqrt{\int_{\mathbf{x}^{(1)},\dots,\mathbf{x}^{(N)}} 1\{\forall n\in[N], \rho_n(\mathbf{x}^{(n)}) > a^{(n)}\} \prod_{n=1}^{N} dq_{\phi^{(n)}}(\mathbf{x}^{(n)})} \tag{18}$$

$$= \sqrt{\mathbb{E}_{X^{(n)}\sim q_{\phi^{(n)}}, \forall n\in[N]}[(\frac{1}{N}\sum_{m=1}^{N} f(X^{(m)}))^2]} \cdot \sqrt{\mathbb{P}(\forall n\in[N], \rho_n(X^{(n)}) > a^{(n)})} \tag{19}$$

$$= ||f||_{\mathbf{q}_\phi} \cdot \sqrt{\mathbb{P}(\forall n\in[N], \rho_n(X^{(n)}) > a^{(n)})}. \tag{20}$$

Similarly,

$$\mathbb{E}|I_K(f) - I_K(h)| = \mathbb{E}\left|\frac{1}{\prod_{n=1}^{N} K^{(n)}} \sum_{k^{(1)}=1}^{K^{(1)}} \cdots \sum_{k^{(N)}=1}^{K^{(N)}} \frac{1}{N}(\sum_{m=1}^{N} f(Y^{(m)}_{[k^{(m)}]}) - h(Y^{(m)}_{[k^{(m)}]})) \prod_{n=1}^{N} \rho_n(Y^{(n)}_{[k^{(n)}]})\right| \tag{21}$$

$$\leq \mathbb{E}\left|\frac{1}{N}(\sum_{m=1}^{N} f(Y^{(m)}_{[k^{(m)}]}) - h(Y^{(m)}_{[k^{(m)}]})) \prod_{n=1}^{N} \rho_n(X^{(n)})\right| \tag{22}$$

$$= \mathbb{E}[|\frac{1}{N}\sum_{m=1}^{N} f(X^{(m)})|; \forall n\in[N], \rho_n(X^{(n)}) > a^{(n)}] \tag{23}$$

$$\leq ||f||_{\mathbf{q}_\phi} \cdot \sqrt{\mathbb{P}(\forall n\in[N], \rho_n(X^{(n)}) > a^{(n)})}. \tag{24}$$

From (23) to (24), we follow the same steps in (16)-(20).

Finally, note that

$$\mathbb{E}|I_K(h) - I(h)| \leq \sqrt{Var(I_K(h))} \tag{25}$$

$$= \sqrt{\frac{1}{\prod_{n=1}^{N} K^{(n)}} Var\left(\frac{1}{N}\sum_{m=1}^{N} h(Y^{(m)}_{[1]}) \cdot \prod_{n=1}^{N} \rho_n(Y^{(n)}_{[1]})\right)} \tag{26}$$

$$\leq \sqrt{\frac{1}{\prod_{n=1}^{N} K^{(n)}} \mathbb{E}\left[(\frac{1}{N}\sum_{m=1}^{N} h(Y^{(n)}_{[1]}))^2 \prod_{n=1}^{N} (\rho_n(Y^{(n)}_{[1]}))^2\right]} \tag{27}$$

$$\leq \sqrt{\frac{\prod_{n=1}^{N} a^{(n)}}{\prod_{n=1}^{N} K^{(n)}} \mathbb{E}\left[(\frac{1}{N} \sum_{m=1}^{N} f(Y_{[1]}^{(m)}))^2 \prod_{n=1}^{N} \rho_n(Y_{[1]}^{(n)}) \right]} \tag{28}$$

$$= ||f||_{\mathbf{q}_\phi} \prod_{n=1}^{N} \left(\frac{a^{(n)}}{K^{(n)}} \right)^{1/2}. \tag{29}$$

Combining the upper bounds above, we get

$$\mathbb{E}\left[|I_K(f) - I(f)|\right] \leq ||f||_{\mathbf{q}_\phi} \left(\prod_{n=1}^{N} \left(\frac{a^{(n)}}{K^{(n)}} \right)^{1/2} + 2\sqrt{\prod_{n=1}^{N} \mathbb{P}\left(\log \rho_n(X^{(n)}) > \log a^{(n)} \right)} \right) \tag{30}$$

$$= ||f||_{\mathbf{q}_\phi} \left(e^{-Nr/4} + 2\sqrt{\prod_{n=1}^{N} \mathbb{P}\left(\log \rho_n(X^{(n)}) > L^{(n)} + r/2 \right)} \right) \tag{31}$$

$$= ||f||_{\mathbf{q}_\phi} \left(e^{-Nr/4} + 2\sqrt{\prod_{n=1}^{N} \mathbb{P}\left(\log \rho_n(X^{(n)}) > D_{KL}(q_{\phi^{(n)}}||p) + r/2 \right)} \right). \tag{32}$$

This completes the proof of the first part of the theorem.

For the converse part, suppose $K^{(n)} = e^{L^{(n)} - r}$ and $a^{(n)} = e^{L^{(n)} - r/2}$ $\forall n \in [N]$. Then,

$$\mathbb{P}(I_K(\mathbf{1}) \geq 1 - \delta) = \mathbb{P}\left(\frac{1}{\prod_{n=1}^{N} K^{(n)}} \sum_{k_1=1}^{K_1} \cdots \sum_{k_N=1}^{K_N} \prod_{n=1}^{N} \rho_n(Y_{[k^{(n)}]}^{(n)}) \geq 1 - \delta \right) \tag{33}$$

$$\begin{aligned} \leq & \mathbb{P}\left(\max_{1 \leq k \leq K^{(n)}} \rho_n(Y_{[k]}^{(n)}) > a^{(n)}, \forall n \in [N] \right) \\ & + \mathbb{P}\left(\frac{1}{\prod_{n=1}^{N} K^{(n)}} \sum_{k^{(1)}=1}^{K^{(1)}} \cdots \sum_{k^{(N)}=1}^{K^{(N)}} \prod_{n=1}^{N} \rho_n(Y_{[k^{(n)}]}^{(n)}) 1\{\forall n \in [N], \rho_n(Y_{[k^{(n)}]}^{(n)}) \leq a^{(n)}\} \geq 1 - \delta \right) \end{aligned} \tag{34}$$

$$\begin{aligned} \leq & \sum_{k^{(1)}=1}^{K^{(1)}} \cdots \sum_{k^{(N)}=1}^{K^{(N)}} \mathbb{P}\left(\rho_n(Y_{[k^{(n)}]}^{(n)}) > a^{(n)}, \forall n \in [N] \right) \\ & + \frac{1}{1-\delta} \mathbb{E}\left[\frac{1}{\prod_{n=1}^{N} K^{(n)}} \sum_{k^{(1)}=1}^{K^{(1)}} \cdots \sum_{k^{(N)}=1}^{K^{(N)}} \prod_{n=1}^{N} \rho_n(Y_{[k^{(n)}]}^{(n)}) 1\{\forall n \in [N], \rho_n(Y_{[k^{(n)}]}^{(n)}) \leq a^{(n)}\} \right] \end{aligned} \tag{35}$$

$$\leq \frac{1}{\prod_{n=1}^{N} a^{(n)}} \sum_{k^{(1)}=1}^{K^{(1)}} \cdots \sum_{k^{(N)}=1}^{K^{(N)}} \prod_{n=1}^{N} \mathbb{E}\left[\rho_n(Y_{[k^{(n)}]}^{(n)}) \right] + \frac{1 - \prod_{n=1}^{N} \mathbb{P}\left(\rho_n(Z) \geq a^{(n)} \right)}{1 - \delta} \tag{36}$$

$$= \prod_{n=1}^{N} \frac{K^{(n)}}{a^{(n)}} + \frac{\prod_{n=1}^{N} \mathbb{P}\left(\rho_n(Z) \leq a^{(n)} \right)}{1 - \delta} \tag{37}$$

$$= e^{-Nr/2} + \frac{\prod_{n=1}^{N} \mathbb{P}\left(\log \rho_n(X^{(n)}) \leq D_{KL}(q_{\phi^{(n)}}||p_\theta) - r/2 \right)}{1 - \delta}, \tag{38}$$

where from (33) to (35) and (34) to (35), we use Markov's inequality. This completes the proof of the second inequality in the theorem statement. □

Now, we restate Theorem 2.1 below and provide the proof afterward.

Theorem C.2 (Theorem 2.1)**.** *Let all notations be as in Theorem C.1 and let $J_K(f)$ be the estimate defined in (9). Suppose that $K^{(n)} = \exp\left(L^{(n)} + r\right)$ for some $r \geq 0$. Let*

$$\epsilon = \left(e^{-Nr/4} + 2\sqrt{\prod_{n=1}^{N} \mathbb{P}(\log \rho_n(X^{(n)}) > L^{(n)} + r/2)} \right)^{1/2}. \tag{39}$$

Then

$$\mathbb{P}\left(|J_K(f) - I(f)| \geq \frac{2||f||_{\mathbf{q}_\phi}\epsilon}{1-\epsilon} \right) \leq 2\epsilon. \tag{40}$$

Proof. Suppose that $K^{(n)} = e^{L^{(n)}+r}$ and $a^{(n)} = e^{L^{(n)}+r/2}\ \forall n \in [N]$. Let

$$b = \sqrt{\prod_{n=1}^{N} \frac{a^{(n)}}{K^{(n)}}} + 2\sqrt{\prod_{n=1}^{N} \mathbb{P}\left(\rho_n(X^{(n)}) > a^{(n)}\right)}. \tag{41}$$

Then, by Theorem C.1, for any $\epsilon, \delta \in (0,1)$,

$$\mathbb{P}\left(|I_K(1) - 1| \geq \epsilon\right) \leq \frac{b}{\epsilon} \tag{42}$$

and

$$\mathbb{P}\left(|I_K(f) - I(f)| \geq \delta\right) \leq \frac{||f||_{\mathbf{q}_\phi} b}{\delta}. \tag{43}$$

Now, if $|I_K(f) - I(f)| < \delta$ and $|I_K(1) - 1| < \epsilon$, then

$$|J_K(f) - I(f)| = \left| \frac{I_K(f)}{I_K(1)} - I(f) \right| \tag{44}$$

$$\leq \frac{|I_K(f) - I(f)| + |I(f)||1 - I_K(1)|}{I_K(1)} \tag{45}$$

$$< \frac{\delta + |I(f)|\epsilon}{1-\epsilon}. \tag{46}$$

Taking $\epsilon = \sqrt{b}$ and $\delta = ||f||_{\mathbf{q}_\phi}\epsilon$ completes the proof of the first inequality in the theorem statement. Note that if ϵ is bigger than 1, the bound is true anyway.

This completes the proof of the theorem. □

D. Examples of KLMS Adaptated to Well-Known Stochastic FL Frameworks

In this section, we provide four concrete examples illustrating how KLMS can be naturally integrated into different FL frameworks with natural choices of pre-data and post-data distributions. Later, in Section 3, we present experimental results showing the empirical improvements KLMS brings in all these cases.

D.1. FedPM-KLMS

As described in Appendix A.1, in FedPM (Isik et al., 2023b), the server holds a global probability mask, which parameterizes a probability distribution over the mask parameters – indicating for each model parameter, with what probability it should remain in the subnetwork. Similarly, each client obtains a local probability mask after local training – parameterizing their locally updated probability assignment for each model parameter to remain in the subnetwork. Choosing the global probability mask $\theta^{(t)}$ as the parameters of the pre-data distribution $p_{\theta^{(t)}}$ and the local probability mask $\phi^{(t,n)}$ as the parameters of the post-data distribution $q_{\phi^{(t,n)}}$ is only natural since the goal in (Isik et al., 2023b) is to send a sample from the local probability distribution $\text{Bern}(\cdot;\phi^{(t,n)})$ with as few bits as possible. This new framework, FedPM-KLMS, provides 50 times reduction in bitrate over vanilla FedPM. The pseudocode for FedPM-KLMS can be found in Algorithm 10.

D.2. QSGD-KLMS

As explained in detail in Appendix A.1, QSGD (Alistarh et al., 2017) is a stochastic quantization method for FL frameworks that train deterministic model parameters, which outperforms many other baselines in the same setting. Focusing on the most extreme case when the number of quantization levels is $s = 1$, we can express the QSGD distribution in (2) as follows:

$$p_{\texttt{QSGD}}(\hat{\mathbf{v}}_i^{(t,n)}) = \begin{cases} \max\left\{\frac{-\mathbf{v}_i^{(t,n)}}{\|\mathbf{v}^{(t,n)}\|}, 0\right\} & \text{if } \hat{\mathbf{v}}_i^{(t,n)} = -\|\mathbf{v}^{(t,n)}\| \\ \max\left\{\frac{\mathbf{v}_i^{(t,n)}}{\|\mathbf{v}^{(t,n)}\|}, 0\right\} & \text{if } \hat{\mathbf{v}}_i^{(t,n)} = \|\mathbf{v}^{(t,n)}\| \\ 1 - \max\left\{\frac{-\mathbf{v}_i^{(t,n)}}{\|\mathbf{v}^{(t,n)}\|}, \frac{\mathbf{v}_i^{(t,n)}}{\|\mathbf{v}^{(t,n)}\|}, 0\right\} & \text{if } \hat{\mathbf{v}}_i^{(t,n)} = 0 \end{cases}, \tag{47}$$

which is again a very natural choice of post-data distribution $q_{\phi^{(t,n)}}$ since vanilla QSGD requires the clients to take a sample from $p_{\texttt{QSGD}}(\cdot)$ in (47) and communicate the deterministic value of that sample to the server. As for the pre-data distribution, exploiting the temporal correlation in FL, we use the empirical frequencies of the historical updates the server received in the previous round. In other words, in every round t, the server records how many clients communicated a negative value (corresponding to $-\|\mathbf{v}^{(t,n)}\|$), a positive value (corresponding to $\|\mathbf{v}^{(t,n)}\|$), or 0 per coordinate, and constructs the pre-data distribution $p_{\theta^{(t)}}$ from these empirical frequencies for the next rounds. This new framework, QSGD-KLMS, yields 12 times reduction in bitrate over vanilla QSGD. The pseudocode for QSGD-KLMS can be found in Algorithm 11.

D.3. SignSGD-KLM

Since SignSGD (Bernstein et al., 2018) is not a stochastic quantizer, we first introduce some stochasticity to the vanilla SignSGD algorithm and then integrate KLMS into it. Instead of mapping the updates to their signs ± 1 deterministically as in vanilla SignSGD, the stochastic version we propose does this mapping by taking a sample from the following SignSGD distribution

$$p_{\texttt{SignSGD}}(\hat{\mathbf{v}}_i^{(t,n)}) = \begin{cases} \text{Sigmoid}(\frac{\mathbf{v}_i^{(t,n)}}{M}) & \text{if } \hat{\mathbf{v}}_i^{(t,n)} = 1 \\ 1 - \text{Sigmoid}(\frac{\mathbf{v}_i^{(t,n)}}{M}) & \text{if } \hat{\mathbf{v}}_i^{(t,n)} = -1 \end{cases}, \tag{48}$$

for some $M > 0$. Instead of taking a sample from $p_{\texttt{SignSGD}}(\cdot)$ and sending the deterministic value of the sample by spending 1 bpp, we can take advantage of the sign symmetry in the model update (about half of the coordinates have positive/negative signs in the update) and reduce the communication cost. For this, we choose $p_{\texttt{SignSGD}}(\cdot)$ in (48) as the post-data distribution $q_{\phi^{(t,n)}}$, and the uniform distribution $U(0.5)$ from the support $\{-1, 1\}$ as the pre-data distribution $p_{\theta^{(t)}}$. This new method, SignSGD-KLMS, achieves higher accuracy than vanilla SignSGD with 60 times smaller bitrate. The pseudocode for SignSGD-KLMS can be found in Algorithm 12.

D.4. SGLD-KLMS

From the Bayesian FL family, we focus on the recent SGLD framework (Vono et al., 2022) as an example since it provides state-of-the-art results. As explained in detail in Section A.1, due to the stochastic formulation of the Bayesian framework, it is natural to choose the local posterior distributions as the post-data distribution $q_{\phi^{(t,n)}}$, and the global posterior distribution at the server as the pre-data distribution $p_{\theta^{(t)}}$. While extending the existing SGLD algorithm (see Section A.1) with KLMS,

Algorithm 10 `FedPM-KLMS`.

Hyperparameters: thresholds to update block locations $\bar{D}_{KL}^{\max}$ and $\bar{D}_{KL}^{\min}$, maximum block size `MAX_BLOCK_SIZE`.
Inputs: number of iterations T, initial block size S, number of samples K, initial number of blocks $M = \lceil \frac{d}{S} \rceil$, target KL divergence D_{KL}^{target}.
Output: random `SEED` and binary mask parameters $\boldsymbol{m}^{(T)}$.

At the server, initialize a random network with weight vector $\boldsymbol{w}^{\text{init}} \in \mathbb{R}^d$ using a random `SEED`, and broadcast it to the clients; initialize the random score vector $\boldsymbol{s}^{(0,g)} \in \mathbb{R}^d$, and compute $\theta^{(0,g)} \leftarrow \text{Sigmoid}(\boldsymbol{s}^{(0,g)})$, Beta priors $\boldsymbol{\alpha}^{(0)} = \boldsymbol{\beta}^{(0)} = \boldsymbol{\lambda}_0$; initialize `UPDATE`←TRUE and the block locations $I_i^{(t)} = [(i-1)S : iS]$ for $i = 1, \dots, M$ and broadcast to the clients.
for $t = 1, \dots, T$ **do**
  Sample a subset $\mathcal{C}_t \subset \{1, \dots, N\}$ of $|\mathcal{C}_t| = C$ clients without replacement.
  On Client Nodes:
  for $c \in \mathcal{C}_t$ **do**
    Compute $\phi^{(t,c)}$ as in `FedPM` in Algorithm 2.
    if `UPDATE` **then**
      $\{k_{[i]}^*\}_{i=1}^M, I^{(t,c)} \leftarrow$ `Adaptive-KLMS`$(\text{Bern}(\theta^{(t,g)}), \text{Bern}(\phi^{(t,c)}), I^{(t)}, D_{KL}^{\text{target}})$ // See Algorithm 7.
      $M \leftarrow \text{length}(I^{(t,c)})$. // New number of blocks.
    else
      $\{k_{[i]}^*\}_{i=1}^M \leftarrow$ `Adaptive-KLMS`$(\text{Bern}(\theta^{(t,g)}), \text{Bern}(\phi^{(t,c)}), I^{(t)}, D_{KL}^{\text{target}})$ // See Algorithm 7.
    end if
    Send $\{k_{[i]}^*\}_{i=1}^M$ with $K \cdot M$ bits and the average KL divergence across blocks $\bar{D}_{KL}^{(t,c)} \leftarrow \frac{1}{M}\sum_{m=1}^M D_{KL}(\text{Bern}(\phi_{[I_m]}^{(t,c)}) \| \text{Bern}(\theta_{[I_m]}^{(t,g)}))$ with 32 bits to the server.
    if UPDATE **then**
      Send $I^{(t,c)}$ with $M \cdot \log_2(\texttt{MAX\_BLOCK\_SIZE})$ bits.
    end if
  end for

  On the Server Node:
  Receive the selected indices $\{k_{[i]}^*\}_{i=1}^M$, and the average KL divergences $\{\bar{D}_{KL}^{(t,c)}\}_{c\in\mathcal{C}_t}$.
  Compute $\bar{D}_{KL}^{(t)} = \frac{1}{C}\sum_{c\in\mathcal{C}_t} \bar{D}_{KL}^{(t,c)}$.
  if `UPDATE` **then**
    $I^{(t)} \leftarrow$ `Aggregate-Block-Locations` $\left(\{I^{(t,c)}\}_{c\in\mathcal{C}_t}\right)$ // See Algorithm 8.
    `UPDATE` = `False`.
  else
    $I^{(t,c)} \leftarrow I^{(t)}$ for all $c \in \mathcal{C}_t$.
    if $\bar{D}_{KL}^{(t)} > \bar{D}_{KL}^{\max}$ **or** $\bar{D}_{KL}^{(t)} < \bar{D}_{KL}^{\min}$ **then** `UPDATE` = `True` **else** `UPDATE` = `False`.
  end if
  for $c \in \mathcal{C}_t$ **do**
    $\{\hat{\boldsymbol{m}}_{[i]}^{(t,c)}\}_{i=1}^M \leftarrow$ `KLMS`-Decoder$(\text{Bern}(\theta^{(t)}), I^{(t,c)}, K)$ // See Algorithm 9.
  end for
  $\theta^{(t)} = \text{BayesAgg}\left(\{\hat{\boldsymbol{m}}^{(t,c)}\}_{c\in\mathcal{C}_t}, t\right)$ // See Algorithm 3.
  Broadcast `UPDATE`, $I^{(t)}$ and $\theta^{(t)}$ to the clients.
end for
Sample $\boldsymbol{m}^{\text{final}} \sim \text{Bern}(\theta^{(T)})$ and return the final model $\dot{\boldsymbol{w}}^{\text{final}} \leftarrow \boldsymbol{m}^{\text{final}} \odot \boldsymbol{w}^{\text{init}}$.

we inject Gaussian noise locally at each client and scale it such that when all the samples are averaged at the server, the aggregate noise sample $\xi^{(t)}$ (see Eq. (3)) is distributed according to $\mathcal{N}(0, \boldsymbol{I}_d)$ (more details in Appendix D). This new framework, `SGLD-KLMS`, provides both accuracy and bitrate gains over `QLSD` (Vono et al., 2022) – the state-of-the-art compression method for Federated `SGLD`. The pseudocode for `SGLD-KLMS` can be found in Algorithm 13.

Algorithm 11 `QSGD-KLMS`.

Hyperparameters: server learning rate η_S, thresholds to update block locations $\bar{D}_{KL}^{\max}$, $\bar{D}_{KL}^{\min}$, maximum block size `MAX_BLOCK_SIZE`.
Inputs: number of iterations T, initial block size S, number of samples K, initial number of blocks $M = \lceil \frac{d}{S} \rceil$, target KL divergence D_{KL}^{target}.
Output: Final model $\boldsymbol{w}^{(T)}$.

At the server, initialize a random network parameters $\boldsymbol{w}^{(0)} \in \mathbb{R}^d$ and broadcast it to the clients; initialize `UPDATE`←TRUE and the block locations $I_i^{(t)} = [(i-1)S : iS]$ for $i = 1, \dots, M$ and broadcast to the clients.
for $t = 1, \dots, T$ **do**
  Sample a subset $\mathcal{C}_t \subset \{1, \dots, N\}$ of $|\mathcal{C}_t| = C$ clients without replacement.
  On Client Nodes:
  for $c \in \mathcal{C}_t$ **do**
    Receive the empirical frequency from the previous round $p_{\theta^{(t)}}$ from the server.
    Compute $\mathbf{v}^{(t,c)}$ as in `QSGD` in Algorithm 4.
    Compute the local post-data distribution $q_{\phi^{(t,c)}}$ with $\mathbf{v}^{(t,c)}$ using $p_{\texttt{QSGD}}(\cdot)$ in (47).
    if `UPDATE` **then**
      $\{k_{[i]}^*\}_{i=1}^M, I^{(t,c)} \leftarrow$ `Adaptive-KLMS`$(p_{\theta^{(t)}}, q_{\phi^{(t,c)}}, I^{(t)}, D_{KL}^{\text{target}})$ // See Algorithm 7.
      $M \leftarrow$ length$(I^{(t,c)})$. // New number of blocks.
    else
      $\{k_{[i]}^*\}_{i=1}^M \leftarrow$ `Adaptive-KLMS`$(p_{\theta^{(t)}}, q_{\phi^{(t,c)}}, I^{(t)}, D_{KL}^{\text{target}})$ // See Algorithm 7.
    end if
    Send $\{k_{[i]}^*\}_{i=1}^M$ with $K \cdot M$ bits and the average KL divergence across blocks $\bar{D}_{KL}^{(t,c)} \leftarrow \frac{1}{M}\sum_{m=1}^M D_{KL}(q_{\phi_{[I_m]}^{(t,c)}} \| p_{\theta_{[I_m]}^{(t,g)}}))$ with 32 bits to the server.
    if UPDATE **then**
      Send $I^{(c)}$ with $M \cdot \log_2($`MAX_BLOCK_SIZE`$)$ bits.
    end if
  end for

  On the Server Node:
  Receive the selected indices $\{k_{[i]}^*\}_{i=1}^M$, and the average KL divergences $\{\bar{D}_{KL}^{(t,c)}\}_{c\in\mathcal{C}_t}$.
  Compute $\bar{D}_{KL}^{(t)} = \frac{1}{C}\sum_{c\in\mathcal{C}_t} \bar{D}_{KL}^{(t,c)}$.
  if `UPDATE` **then**
    $I^{(t,c)} \leftarrow$ `Aggregate-Block-Locations` $\left(\{I^{(t,c)}\}_{c\in\mathcal{C}_t}\right)$ // See Algorithm 8.
    `UPDATE` = `False`.
  else
    $I^{(t,c)} \leftarrow I^{(t)}$ for all $c \in \mathcal{C}_t$.
    if $\bar{D}_{KL}^{(t)} > \bar{D}_{KL}^{\max}$ **or** $\bar{D}_{KL}^{(t)} < \bar{D}_{KL}^{\min}$ **then** `UPDATE` = `True` **else** `UPDATE` = `False`.
  end if
  for $c \in \mathcal{C}_t$ **do**
    $\{\hat{\mathbf{v}}_{[i]}^{(t,c)}\}_{i=1}^M \leftarrow$ `KLMS`-Decoder$(p_{\theta^{(t)}}, I^{(t,c)}, K)$ // See Algorithm 9.
    Construct the empirical frequency $p_{\theta^{(t+1)}}$ from $\{\hat{\mathbf{v}}_{[i]}^{(t,c)}\}_{i=1}^M$.
  end for
  Compute $\boldsymbol{w}^{(t)} = \boldsymbol{w}^{(t-1)} - \eta_S \frac{1}{C}\sum_{c\in\mathcal{C}_t} \hat{\mathbf{v}}^{(t,c)}$.
  Broadcast `UPDATE`, $I^{(t)}$, $\boldsymbol{w}^{(t)}$, and $p_{\theta^{(t)}}$ to the clients.
end for

Algorithm 12 `SignSGD-KLMS`.

Hyperparameters: server learning rate η_S, thresholds to update block locations $\bar{D}_{KL}^{\max}$, $\bar{D}_{KL}^{\min}$, maximum block size `MAX_BLOCK_SIZE`.
Inputs: number of iterations T, initial block size S, number of samples K, initial number of blocks $M = \lceil \frac{d}{S} \rceil$, target KL divergence D_{KL}^{target}.
Output: Final model $\boldsymbol{w}^{(T)}$.

At the server, initialize a random network parameters $\boldsymbol{w}^{(0)} \in \mathbb{R}^d$ and broadcast it to the clients; initialize `UPDATE`←TRUE and the block locations $I_i^{(t)} = [(i-1)S : iS]$ for $i = 1, \ldots, M$ and broadcast to the clients.
for $t = 1, \ldots, T$ **do**
  Sample a subset $\mathcal{C}_t \subset \{1, \ldots, N\}$ of $|\mathcal{C}_t| = C$ clients without replacement.
  On Client Nodes:
  for $c \in \mathcal{C}_t$ **do**
    Compute $\mathbf{v}^{(t,c)}$ as in other standard FL frameworks such as `QSGD` in Algorithm 4.
    Compute the local post-data distribution $q_{\phi^{(t,c)}}$ with $\mathbf{v}^{(t,c)}$ using $p_{\texttt{SignSGD}}(\cdot)$ in (48).
    $p_{\theta^{(t)}} \leftarrow \text{Unif}(0.5)$ over $\{-1, 1\}$.
    if `UPDATE` **then**
      $\{k_{[i]}^*\}_{i=1}^M, I^{(t,c)} \leftarrow$ `Adaptive-KLMS`$(p_{\theta^{(t)}}, q_{\phi^{(t,c)}}, I^{(t)}, D_{KL}^{\text{target}})$ // See Algorithm 7.
      $M \leftarrow \text{length}(I^{(t,c)})$. // New number of blocks.
    else
      $\{k_{[i]}^*\}_{i=1}^M \leftarrow$ `Adaptive-KLMS`$(p_{\theta^{(t)}}, q_{\phi^{(t,c)}}, I^{(t)}, D_{KL}^{\text{target}})$ // See Algorithm 7.
    end if
    Send $\{k_{[i]}^*\}_{i=1}^M$ with $K \cdot M$ bits and the average KL divergence across blocks $\bar{D}_{KL}^{(t,c)} \leftarrow \frac{1}{M}\sum_{m=1}^M D_{KL}(q_{\phi_{[I_m]}^{(t,c)}} \| p_{\theta_{[I_m]}^{(t,g)}}))$ with 32 bits to the server.
    if UPDATE **then**
      Send $I^{(t,c)}$ with $M \cdot \log_2(\texttt{MAX\_BLOCK\_SIZE})$ bits.
    end if
  end for

  On the Server Node:
  Receive the selected indices $\{k_{[i]}^*\}_{i=1}^M$, and the average KL divergences $\{\bar{D}_{KL}^{(t,c)}\}_{c\in\mathcal{C}_t}$.
  Compute $\bar{D}_{KL}^{(t)} = \frac{1}{C}\sum_{c\in\mathcal{C}_t} \bar{D}_{KL}^{(t,c)}$.
  if `UPDATE` **then**
    $I^{(t)} \leftarrow$ `Aggregate-Block-Locations` $\left(\{I^{(t,c)}\}_{c\in\mathcal{C}_t}\right)$ // See Algorithm 8.
    `UPDATE` = `False`.
  else
    $I^{(t,c)} \leftarrow I^{(t)}$ for all $c \in \mathcal{C}_t$.
    if $\bar{D}_{KL}^{(t)} > \bar{D}_{KL}^{\max}$ **or** $\bar{D}_{KL}^{(t)} < \bar{D}_{KL}^{\min}$ **then** `UPDATE` = `True` **else** `UPDATE` = `False`.
  end if
  for $c \in \mathcal{C}_t$ **do**
    $\{\hat{\mathbf{v}}_{[i]}^{(t,c)}\}_{i=1}^M \leftarrow$ `KLMS`-Decoder$(p_{\theta^{(t)}}, I^{(t,c)}, K)$ // See Algorithm 9.
  end for
  Compute $\boldsymbol{w}^{(t)} = \boldsymbol{w}^{(t-1)} - \eta_S \frac{1}{C}\sum_{c\in\mathcal{C}_t} \hat{\mathbf{v}}^{(t,c)}$.
  Broadcast `UPDATE`, $I^{(t)}$ and $\boldsymbol{w}^{(t)}$ to the clients.
end for

Algorithm 13 `SGLD-KLMS`.

Hyperparameters: server learning rate η_S, minibatch size B, thresholds to update block locations $\bar{D}_{KL}^{\max}$, $\bar{D}_{KL}^{\min}$, maximum block size `MAX_BLOCK_SIZE`.
Inputs: number of iterations T, initial block size S, number of samples K, initial number of blocks $M = \lceil \frac{d}{S} \rceil$, target KL divergence D_{KL}^{target}.
Output: samples $\left\{\theta^{(t)}\right\}_{t=1}^{T}$.

At the server, initialize a random network with weight vector $\theta^{(0)} \in \mathbb{R}^d$ and broadcast it to the clients; initialize `UPDATE`←TRUE and the block locations $I_i^{(t)} = [(i-1)S : iS]$ for $i = 1, \ldots, M$ and broadcast to the clients.
for $t = 1, \ldots, T$ **do**
  Sample a subset $\mathcal{C}_t \subset \{1, \ldots, N\}$ of $|\mathcal{C}_t| = C$ clients without replacement.
  On Client Nodes:
  for $c \in \mathcal{C}_t$ **do**
    Receive $\theta^{(t-1)}$ from the server and set $\phi^{(t,c)} \leftarrow \theta^{(t-1)}$.
    Compute a stochastic gradient of the potential $H(\phi^{(t,c)})$ as in `QLSD` in Algorithm 5.
    Set $p_{\theta^{(t)}} \leftarrow \mathcal{N}\left(0, \sqrt{\frac{2}{\gamma C^2}}\boldsymbol{I}_d\right)$.
    Set $q_{\phi^{(t,c)}} \leftarrow \mathcal{N}\left(H(\phi^{(t,c)}), \sqrt{\frac{2}{\gamma C^2}}\boldsymbol{I}_d\right)$.
    if `UPDATE` **then**
      $\{k_{[i]}^*\}_{i=1}^M, I^{(t,c)} \leftarrow$ `Adaptive-KLMS`$(p_{\theta^{(t)}}, q_{\phi^{(t,c)}}, I^{(t)}, D_{KL}^{\text{target}})$ // See Algorithm 7.
      $M \leftarrow \text{length}(I^{(t,c)})$. // New number of blocks.
    else
      $\{k_{[i]}^*\}_{i=1}^M \leftarrow$ `Adaptive-KLMS`$(p_{\theta^{(t)}}, q_{\phi^{(t,c)}}, I^{(t)}, D_{KL}^{\text{target}})$ // See Algorithm 7.
    end if
    Send $\{k_{[i]}^*\}_{i=1}^M$ with $K \cdot M$ bits and the average KL divergence across blocks $\bar{D}_{KL}^{(t,c)} \leftarrow \frac{1}{M}\sum_{m=1}^{M} D_{KL}(q_{\phi_{[I_m]}^{(t,c)}} \| p_{\theta_{[I_m]}^{(t,g)}}))$ with 32 bits to the server.
    if UPDATE **then**
      Send $I^{(t,c)}$ with $M \cdot \log_2(\texttt{MAX\_BLOCK\_SIZE})$ bits.
    end if
  end for

  On the Server Node:
  Receive the selected indices $\{k_{[i]}^*\}_{i=1}^M$, and the average KL divergences $\{\bar{D}_{KL}^{(t,c)}\}_{c\in\mathcal{C}_t}$.
  Compute $\bar{D}_{KL}^{(t)} = \frac{1}{C}\sum_{c\in\mathcal{C}_t} \bar{D}_{KL}^{(t,c)}$.
  if `UPDATE` **then**
    $I^{(t)} \leftarrow$ `Aggregate-Block-Locations` $\left(\{I^{(t,c)}\}_{c\in\mathcal{C}_t}\right)$ // See Algorithm 8.
    `UPDATE` = `False`.
  else
    $I^{(t,c)} \leftarrow I^{(t)}$ for all $c \in \mathcal{C}_t$.
    if $\bar{D}_{KL}^{(t)} > \bar{D}_{KL}^{\max}$ **or** $\bar{D}_{KL}^{(t)} < \bar{D}_{KL}^{\min}$ **then** `UPDATE` = `True` **else** `UPDATE` = `False`.
  end if
  for $c \in \mathcal{C}_t$ **do**
    $\{\hat{H}(\phi_{[i]}^{(t,c)}\}_{i=1}^M \leftarrow$ `KLMS-Decoder`$(p_{\theta^{(t)}}, I^{(t,c)}, K)$ // See Algorithm 9.
  end for
  Compute $\theta^{(t)} = \theta^{(t-1)} - \eta_S \frac{1}{C}\sum_{c\in\mathcal{C}_t} \hat{H}(\phi^{(t,c)})$.
  Broadcast `UPDATE`, $I^{(t)}$ and $\theta^{(t)}$ to the clients.
end for

E. Additional Experimental Details

In Tables 1, 2, and 3, we provide the architectures for all the models used in our experiments, namely `CONV4`, `CONV6`, ResNet-18, and LeNet. In the non-Bayesian experiments, clients performed three local epochs with a batch size of 128 and a local learning rate of 0.1; while in the Bayesian experiments, they performed one local epoch. We conducted our experiments on NVIDIA Titan X GPUs on an internal cluster server, using 1 GPU per run.

Table 1: Architectures for `CONV4` and `CONV6` models used in the experiments.

Model	`CONV-4`	`CONV-6`
Convolutional Layers	64, 64, pool 128, 128, pool	64, 64, pool 128, 128, pool 256, 256, pool
Fully-Connected Layers	256, 256, 10	256, 256, 10

Table 2: ResNet-18 architecture.

Name	**Component**
conv1	3×3 conv, 64 filters. stride 1, BatchNorm
Residual Block 1	$\begin{bmatrix} 3 \times 3 \text{ conv, } 64 \text{ filters} \\ 3 \times 3 \text{ conv, } 64 \text{ filters} \end{bmatrix} \times 2$
Residual Block 2	$\begin{bmatrix} 3 \times 3 \text{ conv, } 128 \text{ filters} \\ 3 \times 3 \text{ conv, } 128 \text{ filters} \end{bmatrix} \times 2$
Residual Block 3	$\begin{bmatrix} 3 \times 3 \text{ conv, } 256 \text{ filters} \\ 3 \times 3 \text{ conv, } 256 \text{ filters} \end{bmatrix} \times 2$
Residual Block 4	$\begin{bmatrix} 3 \times 3 \text{ conv, } 512 \text{ filters} \\ 3 \times 3 \text{ conv, } 512 \text{ filters} \end{bmatrix} \times 2$
Output Layer	4×4 average pool stride 1, fully-connected, softmax

Table 3: LeNet architecture for MNIST experiments.

Name	**Component**
conv1	[5×5 conv, 20 filters, stride 1], ReLU, 2×2 max pool
conv2	[5×5 conv, 50 filters, stride 1], ReLU, 2×2 max pool
Linear	Linear $800 \rightarrow 500$, ReLU
Output Layer	Linear $500 \rightarrow 10$

During non-i.i.d. data split, we choose the size of each client's dataset $|\mathcal{D}^{(n)}| = D_n$ by first uniformly sampling an integer j_n from $\{10, 11, \ldots, 100\}$. Then, a coefficient $\frac{j_n}{\sum_n j_j}$ is computed, representing the size of the local dataset D_n as a fraction of the full training dataset size. Moreover, we impose a maximum number of different labels, or classes, c_{max}, that each client can see. This way, highly unbalanced local datasets are generated.

F. Additional Experimental Results

F.1. Non-i.i.d. Data Split:

For the non-i.i.d. experiments in Figure 3, we only compare against the best of our baselines from the i.i.d. results – namely FedPM, QSGD, DRIVE, EDEN, and DP−REC. As explained in Appendix E, $c_{\max}$ is the maximum number of classes each client can see due to the non-i.i.d. split data. In the experiments in Figure 3, we set $c_{\max} = 4$ for CIFAR-10 and $c_{\max} = 40$ for CIFAR-100; with 20 clients out of 100 participating in each round. In Figure 4, we set $c_{\max} = 2$ for CIFAR-10 and $c_{\max} = 20$ for CIFAR-100; with 10 clients out of 100 clients participating in each round. Figures 3 and 3 show similar gains over the baselines as the i.i.d. experiments in Figure 1; in that, KLMS adaptations provide up to 50 times reduction in the communication cost compared to the baselines with final accuracy as high as the best baseline. This indicates that the statistical heterogeneity level in the data split, while reducing the performance of the underlying training schemes, does not affect the improvement brought by KLMS. We further corroborate this observation with additional experiments in Appendix F.4.2.

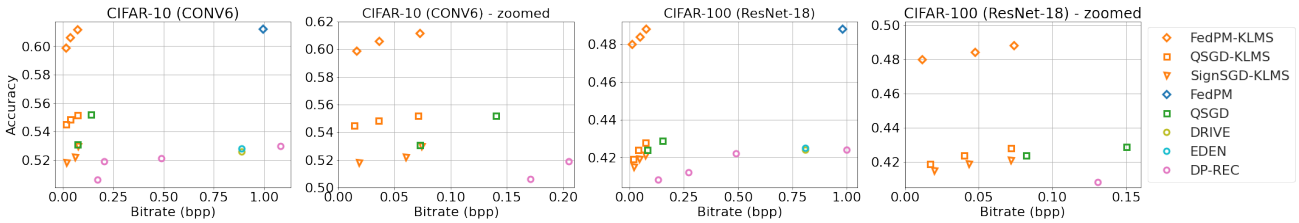

Figure 3: FedPM−KLMS, QSGD−KLMS, and SignSGD−KLMS against FedPM (Isik et al., 2023b), QSGD (Alistarh et al., 2017), DRIVE (Vargaftik et al., 2021), EDEN (Vargaftik et al., 2022), and DP−REC (Liu et al., 2021) with non i.i.d. split and 20 out of 100 clients participating every round.

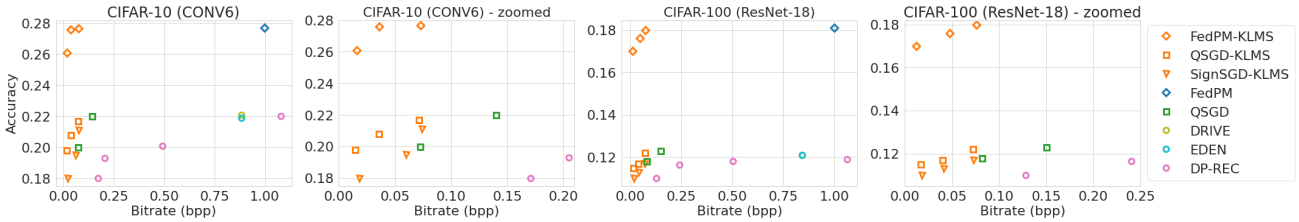

Figure 4: Comparison of FedPM−KLM, QSGD−KLM, and SignSGD−KLM with FedPM (Isik et al., 2023b), QSGD (Alistarh et al., 2017), DRIVE (Vargaftik et al., 2021), EDEN (Vargaftik et al., 2022), and DP−REC (Liu et al., 2021) with non i.i.d. split and 10 out of 100 clients participating every round.

F.2. Bayesian Federated Learning

We present the comparison of SGLD−KLMS with QLSD (Vono et al., 2022) in Figure 5-(left). We consider i.i.d. data split and full client participation with the number of clients $N = 10$. It is seen that SGLD−KLMS can reduce the communication cost by 5 times more than QLSD with higher accuracy on MNIST, where in this case the accuracy is a Monte Carlo average obtained by posterior sampling after convergence.

F.3. Ablation Study: The Effect of the Adaptive Bit Allocation Strategy

We conduct an ablation study to answer the following question: *Does adaptive bit allocation strategy really help optimize the bit allocation and reduce # bits down to KL divergence?* To answer this question, in Figure 5-(right), we show how the average per-parameter KL divergence and # bits spent per parameter change over the rounds for FedPM−KLMS with fixed- and adaptive-size blocks. We adjust the hyperparameters such that the final accuracies differ by only 0.01% on CIFAR-10. For the fixed-size experiments, since we fix K (number of samples per block) and the block size for the whole model and across rounds, # bits per parameter stays the same while the KL divergence shows a decreasing trend. On the other hand, in the adaptive-size experiments, the block size changes across the model parameters and the rounds to guarantee that each block has the same KL divergence. Since all blocks have the same KL divergence, we spend the same # bits for each block as suggested by Theorem 2.1, which adaptively optimizes the bitrate towards the KL divergence. This is indeed

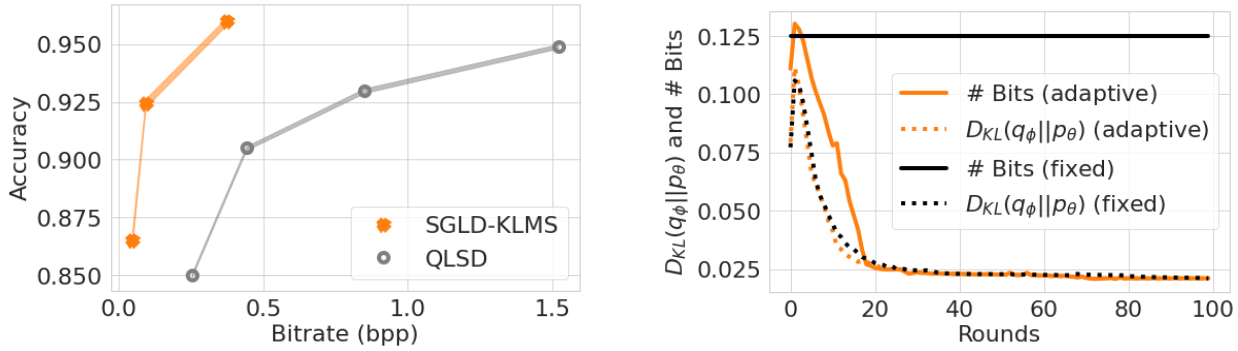

Figure 5: **(left)** `SGLD-KLMS` against `QLSD` (Vono et al., 2022) using LeNet on i.i.d. MNIST dataset. **(right)** `FedPM-KLMS (fixed)` against `FedPM-KLMS (adaptive)` on how well the number of bits approaches the fundamental quantity, KL divergence – using `CONV6` on i.i.d. CIFAR-10. Both KL divergence and the number of bits are normalized by the number of parameters. The final accuracies that `FedPM-KLMS (fixed)` and `FedPM-KLMS (adaptive)` reach differ by only 0.01%.

justified in Figure 5-(right) since the # bits curve quickly approaches the KL divergence curve.

F.4. `KLMS` on a Toy Model

We provide additional insights on `KLMS` employed in a distributed setup similar to that of FL. Specifically, we design a set of experiments in which the server keeps a pre-data distribution $p = \mathcal{N}(0, 1)$, and N clients need to communicate samples according to their local post-data distributions $\{q^{(n)}\}_{n=1}^N = \{\mathcal{N}\left(\mu^{(n)}, 1\right)\}_{n=1}^N$, which are induced by a global and unknown distribution $\mathcal{N}(\mu, 1)$. Each client n applies `KLMS` (see Algorithm 1) to communicate a sample $x^{(n)}$ from $q^{(n)}$ using as coding distribution the pre-data distribution p. The server then computes $\hat{\mu} = \frac{1}{N}\sum_{n=1}^N x^{(n)}$ to estimate μ. We study the effect of N, i.e., the number of clients communicating their samples, on the estimation of μ in different scenarios by varying the rate adopted by the clients (Appenix F.4.1), and the complexity of the problem (Appendix F.4.2).

F.4.1. The effect of the overhead r

In this example, we simulate an i.i.d. data split by having all clients the same local post-data distribution $q^{(n)} = \mathcal{N}(0.8, 1)$ $\forall n \in [N]$. We analyze the bias in the estimation of μ by computing a Monte Carlo average of the discrepancy defined in Section 2.1 (see Figure 7-(**right**)), together with its empirical standard deviation (see Figure 7-(**left**)). From Figure 7, we can observe that, as conjectured, the standard deviation of the gap decreases when N increases, meaning that the estimation is more accurate around its mean value, which is also better for larger values of N. Also, as expected, a larger value of overhead r induces better accuracy.

F.4.2. The effect of non-i.i.d. data split

In this other set of experiments, we simulate a non-i.i.d. data split by inducing, starting from the same pre-data distribution p, different local post-data distributions, simulating drifts in updates statistics due to data heterogeneity. Specifically, we set again $\mu = 0.8$, and then, $\forall n \in [N]$, $\mu^{(n)} = 0.8 + u^{(n)}$, where $u^{(n)} \sim \text{Unif}([-\eta, \eta])$, for $\eta \in \{0.05, 0.1, 0.25, 0.4\}$. In all experiments, $r = 6$. As we can see from the figure, when N is very small (~ 1), a high level of heterogeneity in the update statistics can indeed lead to poor estimation accuracy. However, for reasonable values of N, this effect is considerably mitigated, suggesting that for real-world applications of FL, where the number of devices participating in each round can be very large, `KLMS` can still improve state-of-the-art compression schemes by a large margin, as reported in the results of Section F.1 and Appendix F.5.

F.5. More detailed Results with Confidence Intervals

We now report the confidence intervals for all the experimental results in the paper in Tables 4, 5, 6, 7, 8, 9, 10, and 11 corresponding to Figures 1, 3, and 4.

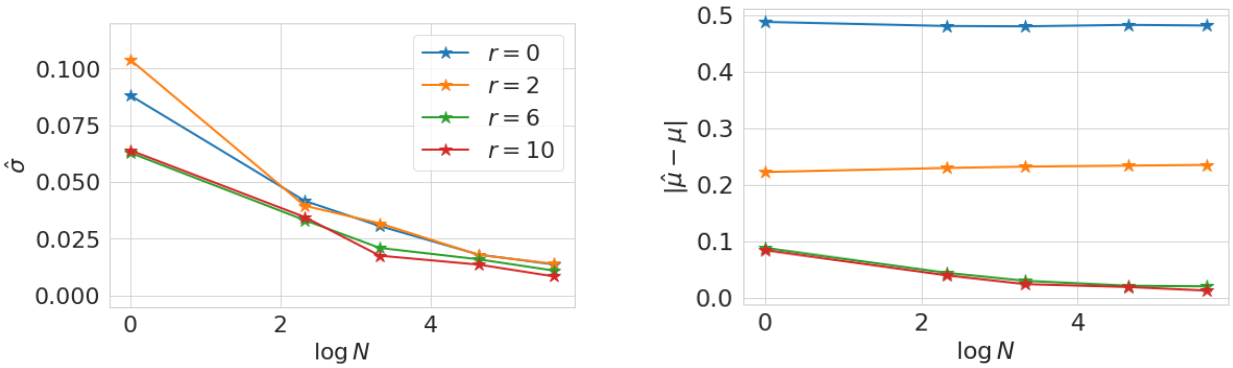

Figure 6: Estimation gap statistics for different values of r, as a function of the number of participating clients N. (**left**) The empirical standard deviation of the estimation gap, computed over 100 runs; (**right**) Estimation gap between μ and $\hat{\mu}$ averaged over 100 runs.

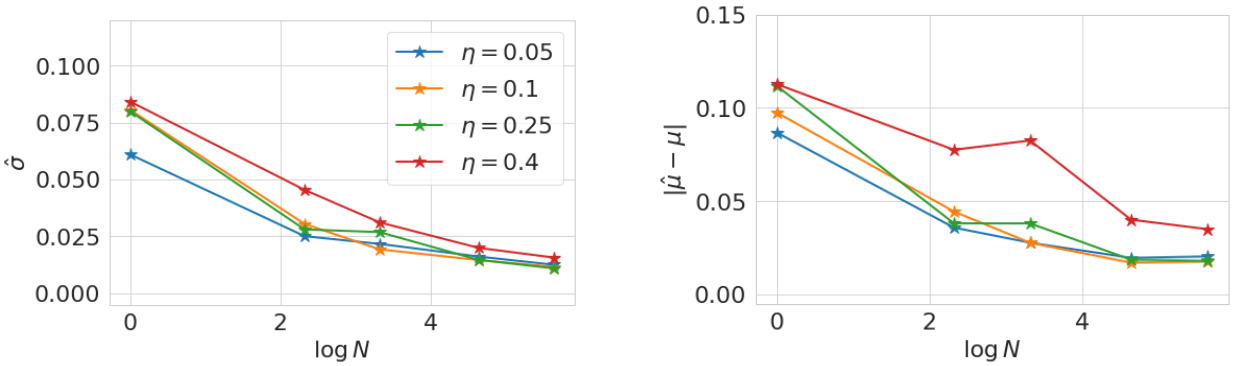

Figure 7: Estimation gap statistics for different values of η, as a function of the number of participating clients N. (**left**) Empirical standard deviation of the estimation gap, computed over 100 runs; (**right**) Estimation gap between μ and $\hat{\mu}$ averaged over 100 runs.

Table 4: Average bitrate $\pm\sigma$ vs final accuracy $\pm\sigma$ in i.i.d. split CIFAR-10 with full client participation. The training duration was set to $t_{\max} = 400$ rounds.

Framework	Bitrate	Accuracy
FedPM-KLMS (ours)	0.070 ± 0.0001	0.787 ± 0.0012
FedPM-KLMS (ours)	0.004 ± 0.0001	0.786 ± 0.0010
FedPM-KLMS (ours)	0.014 ± 0.0001	0.786 ± 0.0012
QSGD-KLMS (ours)	0.071 ± 0.0001	0.765 ± 0.0011
QSGD-KLMS (ours)	0.0355 ± 0.0001	0.761 ± 0.0012
QSGD-KLMS (ours)	0.0142 ± 0.0001	0.755 ± 0.0010
SignSGD-KLMS (ours)	0.072 ± 0.0002	0.745 ± 0.0008
SignSGD-KLMS (ours)	0.040 ± 0.0002	0.745 ± 0.0008
SignSGD-KLMS (ours)	0.015 ± 0.0001	0.739 ± 0.0008
FedPM (Isik et al., 2023b)	0.845 ± 0.0001	0.787 ± 0.0011
QSGD (Alistarh et al., 2017)	0.140 ± 0.0000	0.766 ± 0.0012
QSGD (Alistarh et al., 2017)	0.072 ± 0.0000	0.753 ± 0.0013
SignSGD (Bernstein et al., 2018)	0.993 ± 0.0012	0.705 ± 0.0021
TernGrad (Wen et al., 2017)	1.100 ± 0.0001	0.680 ± 0.0016
DRIVE (Vargaftik et al., 2021)	0.890 ± 0.0000	0.760 ± 0.0010
EDEN (Vargaftik et al., 2022)	0.890 ± 0.0000	0.760 ± 0.0010
FedMask (Li et al., 2021)	1.000 ± 0.0001	0.620 ± 0.0017
DP-REC (Triastcyn et al., 2021)	1.12 ± 0.0001	0.720 ± 0.0011
DP-REC (Triastcyn et al., 2021)	0.451 ± 0.0001	0.690 ± 0.0012
DP-REC (Triastcyn et al., 2021)	0.188 ± 0.0001	0.640 ± 0.0011
DP-REC (Triastcyn et al., 2021)	0.124 ± 0.0001	0.622 ± 0.0013

Table 5: Average bitrate $\pm\sigma$ vs final accuracy $\pm\sigma$ in i.i.d. split CIFAR-100 with full client participation. The training duration was set to $t_{\max} = 400$ rounds.

Framework	Bitrate	Accuracy
FedPM-KLMS (ours)	0.072 ± 0.0001	0.469 ± 0.0010
FedPM-KLMS (ours)	0.040 ± 0.0001	0.461 ± 0.0011
FedPM-KLMS (ours)	0.018 ± 0.0001	0.455 ± 0.0010
QSGD-KLMS (ours)	0.074 ± 0.0001	0.327 ± 0.0010
QSGD-KLMS (ours)	0.043 ± 0.0001	0.319 ± 0.0012
QSGD-KLMS (ours)	0.020 ± 0.0001	0.320 ± 0.0010
SignSGD-KLMS (ours)	0.073 ± 0.0001	0.260 ± 0.0014
SignSGD-KLMS (ours)	0.041 ± 0.0001	0.259 ± 0.0014
SignSGD-KLMS (ours)	0.018 ± 0.0001	0.250 ± 0.0014
FedPM (Isik et al., 2023b)	0.880 ± 0.0001	0.470 ± 0.0010
QSGD (Alistarh et al., 2017)	0.150 ± 0.0000	0.335 ± 0.0011
QSGD (Alistarh et al., 2017)	0.082 ± 0.0000	0.330 ± 0.0011
SignSGD (Bernstein et al., 2018)	0.999 ± 0.0002	0.230 ± 0.0019
TernGrad (Wen et al., 2017)	1.070 ± 0.0001	0.220 ± 0.0015
DRIVE (Vargaftik et al., 2021)	0.540 ± 0.0000	0.320 ± 0.0011
EDEN (Vargaftik et al., 2022)	0.540 ± 0.0000	0.320 ± 0.0010
FedMask (Li et al., 2021)	1.000 ± 0.0001	0.180 ± 0.0014
DP-REC (Triastcyn et al., 2021)	1.06 ± 0.0001	0.280 ± 0.0012
DP-REC (Triastcyn et al., 2021)	0.503 ± 0.0001	0.240 ± 0.0012
DP-REC (Triastcyn et al., 2021)	0.240 ± 0.0001	0.220 ± 0.0012
DP-REC (Triastcyn et al., 2021)	0.128 ± 0.0001	0.170 ± 0.0012

Table 6: Average bitrate $\pm\sigma$ vs final accuracy $\pm\sigma$ in i.i.d. split MNIST with full client participation. The training duration was set to $t_{\max} = 200$ rounds.

Framework	Bitrate	Accuracy
FedPM-KLMS (ours)	0.067 ± 0.0001	0.9945 ± 0.0001
FedPM-KLMS (ours)	0.041 ± 0.0001	0.9945 ± 0.0001
FedPM-KLMS (ours)	0.014 ± 0.0001	0.9943 ± 0.0001
QSGD-KLMS (ours)	0.071 ± 0.0001	0.9940 ± 0.0001
QSGD-KLMS (ours)	0.041 ± 0.0001	0.9938 ± 0.0001
QSGD-KLMS (ours)	0.019 ± 0.0001	0.9935 ± 0.0001
SignSGD-KLMS (ours)	0.0720 ± 0.0001	0.9932 ± 0.0002
SignSGD-KLMS (ours)	0.0415 ± 0.0001	0.9930 ± 0.0002
SignSGD-KLMS (ours)	0.0230 ± 0.0001	0.9918 ± 0.0001
FedPM (Isik et al., 2023b)	0.99 ± 0.0001	0.995 ± 0.0001
QSGD (Alistarh et al., 2017)	0.13 ± 0.0000	0.994 ± 0.0001
QSGD (Alistarh et al., 2017)	0.080 ± 0.0000	0.994 ± 0.0001
SignSGD (Bernstein et al., 2018)	0.999 ± 0.0012	0.990 ± 0.0004
TernGrad (Wen et al., 2017)	1.05 ± 0.0001	0.980 ± 0.0003
DRIVE (Vargaftik et al., 2021)	0.91 ± 0.0000	0.994 ± 0.0001
EDEN (Vargaftik et al., 2022)	0.91 ± 0.0000	0.994 ± 0.0001
FedMask (Li et al., 2021)	1.0 ± 0.0001	0.991 ± 0.0003
DP-REC (Triastcyn et al., 2021)	0.996 ± 0.0001	0.991 ± 0.0001
DP-REC (Triastcyn et al., 2021)	0.542 ± 0.0001	0.989 ± 0.0001
DP-REC (Triastcyn et al., 2021)	0.191 ± 0.0001	0.988 ± 0.0001
DP-REC (Triastcyn et al., 2021)	0.125 ± 0.0001	0.985 ± 0.0001

Table 7: Average bitrate $\pm\sigma$ vs final accuracy $\pm\sigma$ in i.i.d. split EMNIST with full client participation. The training duration was set to $t_{\max} = 200$ rounds.

Framework	Bitrate	Accuracy
FedPM-KLMS (ours)	0.068 ± 0.0001	0.889 ± 0.0001
FedPM-KLMS (ours)	0.034 ± 0.0001	0.888 ± 0.0001
FedPM-KLMS (ours)	0.017 ± 0.0001	0.885 ± 0.0001
QSGD-KLMS (ours)	0.072 ± 0.0001	0.884 ± 0.0001
QSGD-KLMS (ours)	0.042 ± 0.0001	0.884 ± 0.0001
QSGD-KLMS (ours)	0.022 ± 0.0001	0.883 ± 0.0001
SignSGD-KLMS (ours)	0.072 ± 0.0001	0.881 ± 0.0003
SignSGD-KLMS (ours)	0.044 ± 0.0001	0.880 ± 0.0003
SignSGD-KLMS (ours)	0.025 ± 0.0001	0.875 ± 0.0003
FedPM (Isik et al., 2023b)	0.890 ± 0.0001	0.890 ± 0.0001
QSGD (Alistarh et al., 2017)	0.150 ± 0.0000	0.884 ± 0.0001
QSGD (Alistarh et al., 2017)	0.086 ± 0.0000	0.882 ± 0.0001
SignSGD (Bernstein et al., 2018)	1.0 ± 0.0001	0.873 ± 0.0005
TernGrad (Wen et al., 2017)	1.1 ± 0.0001	0.870 ± 0.0005
DRIVE (Vargaftik et al., 2021)	0.9 ± 0.0001	0.8835 ± 0.0001
EDEN (Vargaftik et al., 2022)	0.9 ± 0.0001	0.8835 ± 0.0001
FedMask (Li et al., 2021)	1.0 ± 0.0001	0.862 ± 0.0005
DP-REC (Triastcyn et al., 2021)	1.100 ± 0.0001	0.885 ± 0.0001
DP-REC (Triastcyn et al., 2021)	0.488 ± 0.0001	0.880 ± 0.0001
DP-REC (Triastcyn et al., 2021)	0.196 ± 0.0001	0.873 ± 0.0001
DP-REC (Triastcyn et al., 2021)	0.119 ± 0.0001	0.861 ± 0.0001

Table 8: Average bitrate $\pm\sigma$ vs final accuracy $\pm\sigma$ in non-IID split CIFAR-10 with $c_{\max} = 2$, and partial participation with 10 out of 100 clients participating every round. The training duration was set to $t_{\max} = 200$ rounds.

Framework	Bitrate	Accuracy
FedPM-KLMS (ours)	0.073 ± 0.0001	0.277 ± 0.0005
FedPM-KLMS (ours)	0.036 ± 0.0001	0.276 ± 0.0005
FedPM-KLMS (ours)	0.0161 ± 0.0001	0.261 ± 0.0004
QSGD-KLMS (ours)	0.071 ± 0.0001	0.277 ± 0.0005
QSGD-KLMS (ours)	0.036 ± 0.0001	0.208 ± 0.0005
QSGD-KLMS (ours)	0.014 ± 0.0001	0.198 ± 0.0005
SignSGD-KLMS (ours)	0.074 ± 0.0001	0.211 ± 0.0009
SignSGD-KLMS (ours)	0.060 ± 0.0001	0.195 ± 0.0008
SignSGD-KLMS (ours)	0.018 ± 0.0001	0.180 ± 0.0009
FedPM (Isik et al., 2023b)	0.997 ± 0.0001	0.277 ± 0.0006
QSGD (Alistarh et al., 2017)	0.140 ± 0.0000	0.220 ± 0.0005
QSGD (Alistarh et al., 2017)	0.072 ± 0.0000	0.200 ± 0.0005
DRIVE (Vargaftik et al., 2021)	0.885 ± 0.0000	0.221 ± 0.0005
EDEN (Vargaftik et al., 2022)	0.885 ± 0.0000	0.219 ± 0.0004
DP-REC (Triastcyn et al., 2021)	1.080 ± 0.0001	0.220 ± 0.0007
DP-REC (Triastcyn et al., 2021)	0.490 ± 0.0001	0.201 ± 0.0006
DP-REC (Triastcyn et al., 2021)	0.205 ± 0.0001	0.193 ± 0.0006
DP-REC (Triastcyn et al., 2021)	0.171 ± 0.0001	0.180 ± 0.0006

Table 9: Average bitrate $\pm\sigma$ vs final accuracy $\pm\sigma$ in non-IID split CIFAR-10 with $c_{\max} = 4$, and partial participation with 20 out of 100 clients participating every round. The training duration was set to $t_{\max} = 200$ rounds.

Framework	Bitrate	Accuracy
FedPM-KLMS (ours)	0.073 ± 0.0001	0.612 ± 0.0010
FedPM-KLMS (ours)	0.036 ± 0.0001	0.606 ± 0.0010
FedPM-KLMS (ours)	0.016 ± 0.0001	0.599 ± 0.0010
QSGD-KLMS (ours)	0.071 ± 0.0001	0.552 ± 0.0010
QSGD-KLMS (ours)	0.036 ± 0.0001	0.549 ± 0.0011
QSGD-KLMS (ours)	0.014 ± 0.0001	0.545 ± 0.0010
SignSGD-KLMS (ours)	0.074 ± 0.0001	0.530 ± 0.0013
SignSGD-KLMS (ours)	0.060 ± 0.0001	0.522 ± 0.0013
SignSGD-KLMS (ours)	0.018 ± 0.0001	0.518 ± 0.0013
FedPM (Isik et al., 2023b)	0.993 ± 0.0001	0.612 ± 0.0009
QSGD (Alistarh et al., 2017)	0.140 ± 0.0000	0.552 ± 0.0010
QSGD (Alistarh et al., 2017)	0.072 ± 0.0000	0.531 ± 0.0010
DRIVE (Vargaftik et al., 2021)	0.888 ± 0.0000	0.526 ± 0.0010
EDEN (Vargaftik et al., 2022)	0.888 ± 0.0000	0.528 ± 0.0010
DP-REC (Triastcyn et al., 2021)	1.080 ± 0.0001	0.530 ± 0.0012
DP-REC (Triastcyn et al., 2021)	0.490 ± 0.0001	0.521 ± 0.0012
DP-REC (Triastcyn et al., 2021)	0.205 ± 0.0001	0.519 ± 0.0012
DP-REC (Triastcyn et al., 2021)	0.171 ± 0.0001	0.506 ± 0.0012

Table 10: Average bitrate $\pm\sigma$ vs final accuracy $\pm\sigma$ in non-IID split CIFAR-100 with $c_{\max} = 20$, and partial participation with 10 out of 100 clients participating every round. The training duration was set to $t_{\max} = 200$ rounds.

Framework	Bitrate	Accuracy
FedPM-KLMS (ours)	0.076 ± 0.0001	0.180 ± 0.0012
FedPM-KLMS (ours)	0.048 ± 0.00101	0.176 ± 0.0011
FedPM-KLMS (ours)	0.012 ± 0.0001	0.170 ± 0.0011
QSGD-KLMS (ours)	0.072 ± 0.0001	0.122 ± 0.0012
QSGD-KLMS (ours)	0.040 ± 0.0001	0.117 ± 0.0012
QSGD-KLMS (ours)	0.017 ± 0.0001	0.115 ± 0.0012
SignSGD-KLMS (ours)	0.073 ± 0.0001	0.117 ± 0.0014
SignSGD-KLMS (ours)	0.041 ± 0.0001	0.113 ± 0.0014
SignSGD-KLMS (ours)	0.018 ± 0.0001	0.110 ± 0.0013
FedPM (Isik et al., 2023b)	0.999 ± 0.0001	0.181 ± 0.0011
QSGD (Alistarh et al., 2017)	0.150 ± 0.0000	0.123 ± 0.0012
QSGD (Alistarh et al., 2017)	0.082 ± 0.0000	0.118 ± 0.0012
DRIVE (Vargaftik et al., 2021)	0.840 ± 0.0000	0.121 ± 0.0012
EDEN (Vargaftik et al., 2022)	0.840 ± 0.0000	0.121 ± 0.0012
DP-REC (Triastcyn et al., 2021)	1.060 ± 0.0001	0.119 ± 0.0012
DP-REC (Triastcyn et al., 2021)	0.503 ± 0.0001	0.118 ± 0.0013
DP-REC (Triastcyn et al., 2021)	0.240 ± 0.0001	0.117 ± 0.0013
DP-REC (Triastcyn et al., 2021)	0.128 ± 0.0001	0.110 ± 0.0013

Table 11: Average bitrate $\pm\sigma$ vs final accuracy $\pm\sigma$ in non-IID split CIFAR-100 with $c_{\max} = 40$, and partial participation with 20 out of 100 clients participating every round. The training duration was set to $t_{\max} = 200$ rounds.

Framework	Bitrate	Accuracy
FedPM-KLMS (ours)	0.074 ± 0.0001	0.488 ± 0.0013
FedPM-KLMS (ours)	0.048 ± 0.0001	0.484 ± 0.0013
FedPM-KLMS (ours)	0.012 ± 0.0001	0.480 ± 0.0013
QSGD-KLMS (ours)	0.072 ± 0.0001	0.428 ± 0.0013
QSGD-KLMS (ours)	0.040 ± 0.0001	0.424 ± 0.0013
QSGD-KLMS (ours)	0.017 ± 0.0001	0.419 ± 0.0013
SignSGD-KLMS (ours)	0.072 ± 0.0001	0.421 ± 0.0016
SignSGD-KLMS (ours)	0.044 ± 0.0001	0.419 ± 0.0016
SignSGD-KLMS (ours)	0.020 ± 0.0001	0.415 ± 0.0016
FedPM (Isik et al., 2023b)	0.980 ± 0.0001	0.488 ± 0.0012
QSGD (Alistarh et al., 2017)	0.150 ± 0.0000	0.429 ± 0.0013
QSGD (Alistarh et al., 2017)	0.082 ± 0.0000	0.424 ± 0.0013
DRIVE (Vargaftik et al., 2021)	0.81 ± 0.0000	0.424 ± 0.0013
EDEN (Vargaftik et al., 2022)	0.81 ± 0.0000	0.425 ± 0.0013
DP-REC (Triastcyn et al., 2021)	1.00 ± 0.0001	0.424 ± 0.0014
DP-REC (Triastcyn et al., 2021)	0.49 ± 0.0001	0.422 ± 0.0014
DP-REC (Triastcyn et al., 2021)	0.27 ± 0.0001	0.412 ± 0.0014
DP-REC (Triastcyn et al., 2021)	0.13 ± 0.0001	0.408 ± 0.0014