Federated Fine-Tuning of Vision Foundation Models via Probabilistic Masking

Vasileios Tsouvalas [1] Yuki M Asano [2] Aaqib Saeed [1]

Abstract

Foundation Models (FMs) have revolutionized machine learning with their adaptability and high performance across tasks; yet, their integration into Federated Learning (FL) is challenging due to substantial communication overhead from their extensive parameterization. We present `DeltaMask`, a novel method that efficiently fine-tunes FMs in FL at an ultra-low bitrate, well below 1 bpp. `DeltaMask` employs stochastic masking to detect highly effective subnetworks within FMs and leverage stochasticity and sparsity in client masks to compress updates into a compact grayscale image using probabilistic filters, deviating from traditional weight training approaches. Our comprehensive evaluations across various datasets and architectures demonstrate `DeltaMask` efficiently achieves bitrates as low as 0.09 bpp, enhancing communication efficiency while maintaining FMs performance, as measured on 8 datasets and 5 pre-trained models of various network architectures.

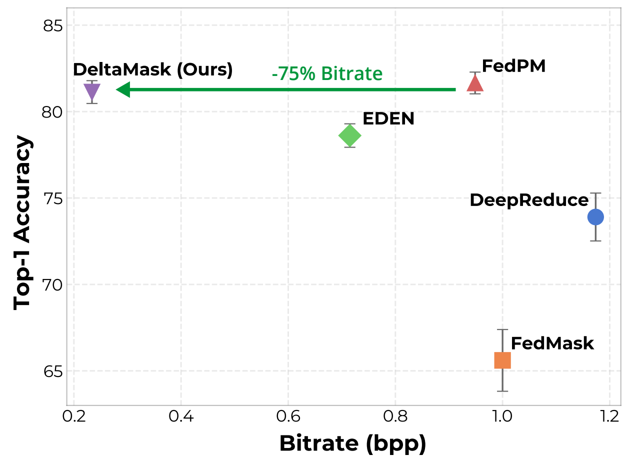


Figure 1. `DeltaMask` (Ours) vs. state-of-the-art communication-efficient FL techniques with pre-trained CLIP ViT-B/32.

1. Introduction

Federated learning (FL) enables collaborative training of neural network models directly on edge devices (referred to as clients), locally with on-device data (Konečný et al., 2016). Despite its appealing properties for users' privacy, FL requires constant models' updates transfer between server and clients, which poses a challenge in terms of communication efficiency. This becomes even more critical when the clients are resource-constrained edge devices, which operate under limited transmission bandwidth and strict energy constraints. Recent advances in FL have led to a variety of methods aimed at enhancing communication efficiency, particularly by reducing the data volume exchanged in each federated round. These strategies often employ gradient compression techniques, including sparsification (Lin et al., 2020; Aji & Heafield, 2017), quantization (Alistarh et al., 2017; Vargaftik et al., 2022; 2021), and low-rank approximation (Mohtashami et al., 2022; Mozaffari et al., 2022), which are pivotal in streamlining data transmission.

Similarly, the "Lottery Ticket Hypothesis" (Frankle & Carbin, 2019) has paved the way for FL training regimes that diverge from traditional weight updates. Here, the focus has shifted toward identifying and cultivating high-potential subnetworks within randomly initialized neural models (Li et al., 2021; Vallapuram et al., 2022; Li et al., 2020; Isik et al., 2023). Such subnetworks demonstrate good performance without the need for extensive weight adjustments, offering a viable path to minimize FL communication overhead. FedMask (Li et al., 2021) and FedPM (Isik et al., 2023), which learn binary masks on top of random dense networks, are shown to reduce bitrate from 32 to 1 bit-per-parameter (bpp). However, jointly learning effective subnetworks in large, randomly initialized models, severely affect training duration and model convergence.

Leveraging the advancements in self-supervised learning, vision Foundation Models (FMs) have brought significant advancement across various machine learning domains with their remarkable representation quality. Models like CLIP (Radford et al., 2021) and DINOv2 (Oquab et al., 2023) demonstrate rapid adaptability to diverse tasks, achieving unmatched performance in several downstream applications. Notably, recent developments have seen vision FMs, such as the ViT models, expand to billions of

[1]Eindhoven University of Technology [2]University of Amsterdam. Correspondence to: Vasileios Tsouvalas <v.tsouvalas@tue.nl>.

Published at ICML 2024 Workshop on Foundation Models in the Wild.

parameters (Dehghani et al., 2023), exemplifying the scale and complexity of modern FMs. In turn, and as an alternative to traditional fine-tuning, masking strategies have emerged (Mallya et al., 2018; Zhao et al., 2020) in a centralized setting, where selective binary masks are learned on top of frozen pre-trained weights, matching the performance of full fine-tuning. Nevertheless, the high parameter count of FMs inhibits their straightforward expansion into decentralized settings due to the substantial communication overhead (Zhuang et al., 2023), even at a bitrate of 1 bpp, thereby limiting their potential to tap into the valuable data available in distributed environments.

To bridge the gap between the high-performance potential of Foundation Models (FMs) and the practical constraints of federated settings, we introduce `DeltaMask`, an approach designed to fine-tune FMs to various downstream tasks in federated settings with significantly reduced bitrate requirements (see Fig.1). Inspired by the sparse mask updates between subsequent federated rounds, which naturally occur due to the rapid adaptability of FMs, our approach combines stochastic masking with probabilistic filters to find high-performing subnetworks within pre-trained FM, while operating in an ultra-low bitrate regime. This paves the way for fine-tuning FMs in federated settings without the massive communication burden caused by their large number of parameters, crucial for scenarios where niche, sensitive data must remain local yet is immensely valuable for learning, such as in healthcare applications. Concisely, the main contributions of our work are as follows:

- We present a simple, yet effective, method termed `DeltaMask`, to fine-tune FMs in FL in a highly communication-efficient manner.
- We combine stochastic binary masks with probabilistic filters to compactly communicate mask updates and reduce bitrate bpp below 0.1.
- Our evaluation across 8 datasets and 5 pre-trained models demonstrate `DeltaMask`'s effectiveness to fine-tune FMs compared to existing FL techniques.

2. Methodology

Overview. We present the general `DeltaMask` training pipeline. First, clients initialize a neural network $p_{w_{\text{init}}}$ with the weight vector $w_{\text{init}} \in \mathbb{R}^d$, from the pre-trained foundation model. The weight vector w_{init} is kept fixed and never modified during training. `DeltaMask` collaboratively trains a probabilistic mask $\theta \in [0,1]^d$, such that the function $\mathcal{L}_{\dot{w}}$ minimize its error rate on a given downstream task ($\dot{w} = m \odot w_{\text{init}}$). In every round t, the server samples a set K_{t} participants ($|K_{\text{t}}|$=K out of N clients), who train their local probability masks $\theta^{\text{k},t}$ on their locally stored datasets D^{k} (of $|D_{\text{k}}|$ samples).

Instead of communicating the stochastic binary mask m, we reduce the required bits-per-parameter (bpp) by only communicating key updates (position indexes set Δ) between the received and trained mask. We represent Δ using probabilistic filters (see Appx. B for formal definition) and transmit the fingerprint set $\mathcal{H}$ (hashed filter's entries) to the server as a single gray-scaled image. On server-side, reconstruction of client masks $m^{\text{k},t}$ is feasible via fast membership checks using the probabilistic filter. The server then aggregates these local masks to complete the t_{th} round.

Compressing Mask Updates. Our approach utilizes a local training scheme for probability masks, where clients aim to learn a binary mask via stochastic mask training. In brief, clients receive a global probability mask $\theta^{\text{g},t-1}$ at round t, where each client k performs local training and updates the mask via back-propagation. To satisfy $\theta^{\text{k},t} \in [0,1]^d$ without clipping, we apply a sigmoid operation over the mask's *unbounded* mask scores $s^{\text{k},t} \in \mathbb{R}^d$. Then, clients can utilize a binary mask $m^{\text{k},t}$, i.e, sampled from Bern($\theta^{\text{k},t}$) and aim to minimize $\mathcal{L}(p_{\dot{w}_{\text{k},t}}, D^{\text{k}})$ over their locally stored data, D^{k}, after which they back-propagate to update $s^{\text{k},t}$.

To enable ultra-low bitrate levels, `DeltaMask` leverages the inherent sparsity in consecutive mask updates across rounds. For a given round t, we deterministically sample a binary mask $m^{\text{g},t-1}$ from the received mask distribution, Bern($\theta^{\text{g},t-1}$), using a publicly shared *seed*. This ensures uniformity of the binary mask among all clients ($m_i^{\text{g},t-1} = m_j^{\text{g},t-1}$ for any $i, j \in K$). Instead of communicating $m^{\text{k},t}$, we catalog the index positions of differences between $m^{\text{g},t-1}$ and $m^{\text{k},t}$, creating a set of index differences, $\Delta^{\text{k},t}$. As mask update sparsity increases during training, we introduce a top_κ ranking that selects $\kappa\%$ of $\Delta^{\text{k},t}$ based on their relative entropy between $m^{\text{g},t-1}$ and $m^{\text{k},t}$. This introduces importance sampling into the communication scheme, similar to (Chatterjee & Diaconis, 2017; Havasi et al., 2018), minimizing distributed mean estimation error by providing essential updates early in training and conveying detailed information of $m^{\text{k},t}$ later on without significantly increasing bitrate due to the growing sparsity of subsequent mask differences. Using Kullback–Leibler (KL) divergence as a measure of entropy, $\Delta^{\text{k},t}$ is defined as:

$$\Delta^{\text{k},t} = \underset{\text{KL}(\theta^{\text{k},t}, \theta^{\text{g},t-1})}{Sort} \left\{ i \,|\, m_i^{\text{g},t-1} \neq m_i^{\text{k},t}, \forall i \in d \right\} [1:\mathsf{K}], \tag{1}$$

where d is the dimension of the probability mask m, and K represents the number of elements to be retained in the sorted set, determined as $\kappa\%$ of $|D_{\text{k}}|$.

Next, we use a *binary fuse filter* with 8 bit-per-entry (*bpe*) to extract a fingerprint array $\mathcal{H}^{\text{k},t}$ from $\Delta^{\text{k},t}$ (see Eq. 3), transitioning from 32-bit indexes to $\approx$ 8-bit hashed entries. We further encode $\mathcal{H}^{\text{k},t}$ into a "*pseudo gray-scale*" image using lossless PNG-like compression, chosen for its wide availability on edge devices, which leverages non-uniform distributions of entries in $\mathcal{H}$ to reduce the bitrate. The resulting image, $\mathrm{A}_{\mathrm{k},t}$, efficiently encapsulates mask updates in a visual, compressed format suitable for server transmission.

Bayesian Aggregation of Compressed Masks. After local training at round t, the server creates a new global probability mask from the received clients' gray-scale images, $\mathrm{A}\mathrm{k}, t$. Specifically, for each client k, the server decompresses $\mathrm{A}\mathrm{k}, t$ to extract $\mathcal{H}^{\mathrm{k},t}$, then uses it to estimate the set of indexes ($\hat{\Delta}$) in $m^{\mathrm{g},t-1}$ that need updating via a membership query across all possible indexes of $m^{\mathrm{g},t-1}$, as follows:

$$\hat{\Delta}^{\mathrm{k},t} = \{i \mid \text{Member}\,(i) = \text{true}, \forall i \in d\} \tag{2}$$

The clients' stochastic binary sample mask $m^{\mathrm{k},t}$ can be constructed by a simple "bit-flip" of $m^{\mathrm{g},t-1}$ in the positions derived from $\hat{\Delta}^{\mathrm{k},t}$. The server can then compute the estimated aggregated probability mask $\bar{\theta}^{\mathrm{g},t} = \frac{1}{K}\sum_{k\in K} m^{\mathrm{k},t}$, which is an unbiased estimation of the underlying probability mask $\theta^{\mathrm{g},t} = \frac{1}{K}\sum_{k\in K} \theta^{\mathrm{k},t}$ using (Ferreira et al.) or a similar strategy (see Appx. C). In contrast to FedPM (Isik et al., 2023) and HideNseek (Vallapuram et al., 2022), our method enables better control over the model generalization versus bitrate during FL training stage by adjusting the *bpe* of the *binary fuse filters* (see Fig.4b). Furthermore, DeltaMask benefits from a bounded estimation error (see Appx. D). Our algorithm can be found in Appx. A.

3. Experiments

Datasets and FL Settings. We conduct experiments across 8 diverse image classification datasets. Note that we focus solely on classification tasks in this work to directly compare with similar approaches (Kostopoulou et al., 2021; Isik et al., 2023; Li et al., 2021; Vargaftik et al., 2022; 2021); yet DeltaMask poses no restriction on the underlying downstream task. We utilize popular ViT architectures pretrained in self-supervised manner, such as CLIP (Radford et al., 2021), DINOv2 (Oquab et al., 2023), where, we learn a mask only for the last 5 transformer blocks, similar to (Zhao et al., 2020). We use a cosine scheduler for top_κ mechanism starting from κ=0.8. Due to limited space, our full experimental setup can be seen in Appx. F.1.

Baselines. We evaluate DeltaMask in terms of accuracy, bitrate (bits-per-parameters), computational complexity and total volume of communicated data between client and server. From the domain of gradient compression techniques, we incorporate EDEN (Vargaftik et al., 2022), DRIVE (Vargaftik et al., 2021), QSGD (Alistarh et al., 2017), and FedCode (Khalilian et al., 2023) into our evaluation. Additionally, we consider DeepReduce (Kostopoulou et al., 2021) as a baseline owing to its analogous use of bloom filter-based compressor. From the threshold-based masking strategies in FL, we compare DeltaMaskagainst threshold-based masking strategies in FL, including FedMask (Li et al., 2021) (without initial pruning) and FedPM (Isik et al., 2023), which leverages stochastic masking concepts. We use a fixed number of rounds across all baselines to facilitate a direct comparison of data transfer volumes for fine-tuning FMs, as inferred from the reported bitrates (lower is better).

3.1. Results

Bitrate-Accuracy Trade-Off. We focus on non-IID data distribution among clients using $Dir(0.1)$ over classes ($C_p \approx 0.2$). Furthermore, the number of clients (N) set to 30, while we consider partial participation with ρ=0.2 (meaning that in each round $\rho \cdot N = 6$ clients are randomly selected). As depicted in Fig.2, DeltaMask achieves significant reductions in communication costs compared to the considered baselines — consistently across all datasets. Among the baselines, EDEN requires the least bandwidth, while FedPM attains the highest accuracy; nevertheless, DeltaMask reliably matches the accuracy of FedPM with significant bitrate reduction. This notable improvement in bitrate over FedPM indicates that mask updates entail significant overhead. Transmitting only essential information via binary fuse filters leads to considerable reductions in bpp (up to 9$\times$ less) without compromising on model accuracy. Compared to DeepReduce — which utilizes Bloom filters to transmit the updates — our method underscores the importance of accurate mask reconstruction, as Bloom filters are prone to a higher false positive rate for the same number of hash functions and bits per entry. Due to limited space, we provide additional experiments across various models and federated settings in Appx. F.2-F.5.

Data Volume & Computational Cost Improvements. We evaluate DeltaMask's impact on computational resources at both client and server levels, and its communication efficiency relative to total transmitted data. Using CLIP on CIFAR-100 with N=10, we measure encoding and decoding times for various gradient compression schemes. For FedMask and FedPM, we omit arithmetic encoding to ensure comparable execution times. All tests are conducted on a CPU, excluding aggregation in decoding time measurements. Data volume is normalized to full fine-tuning size, reporting the volume needed to reach within 1% of peak accuracy, to illustrate both communication efficiency and convergence speed analysis.

Among the methods in Fig.3, FedCode is the most communication-efficient in data volume; yet exhibiting the longest encoding times and the lowest model performance. DeepReduce, using a Bloom-based compressor, struggles with scalability due to longer execution times. In contrast,DeltaMask offers significant improvements in filter construction and query times. FedMask and FedPM balance data volume and execution time, with FedPM leading in accuracy. Surprisingly, DeltaMask, while using slightly more data than FedCode, provides quicker encoding, crucial for resource-limited devices, and matches FedPM's high

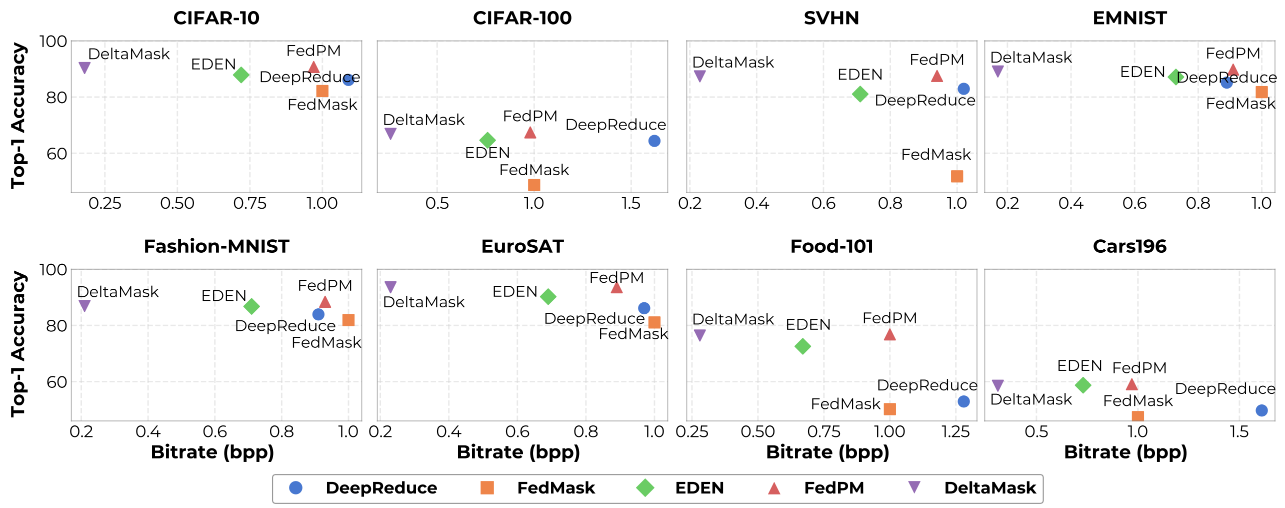
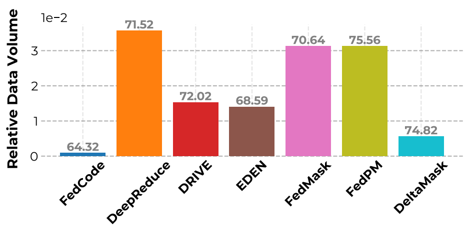
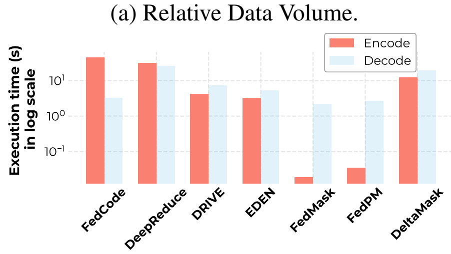


Figure 2. Evaluation of **`DeltaMask` (Ours)** in terms of average bitrate (bits-per-parameter) during FL training using $Dir(0.1)$ over classes ($C_p \approx 0.2$ / **non-IID settings**) for CLIP ViT-B/32. Federated parameters are set to N=30, R=300, ρ=0.2, and E=1.



(a) Relative Data Volume.

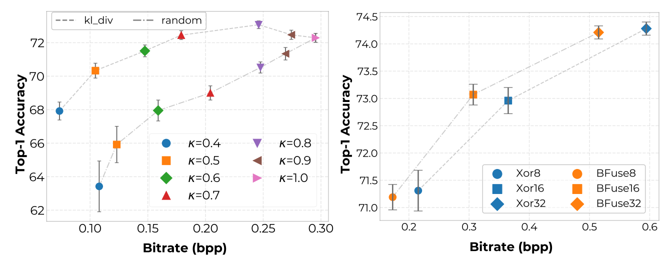


(b) Encode/Decode time (CPU).

Figure 3. Evaluation of **`DeltaMask` (Ours)** in terms of (a) data volume and (b) encoding/decoding time (with baselines), required to reach 1% of peak performance of CLIP ViT-B/32 on CIFAR-100. Data volume normalized over full fine-tuning data size.

accuracy with significantly less communicated data. This makes `DeltaMask` an effective choice for environments with computational and communication constraints. An in-depth analysis on common edge devices is in Appx. F.6.

Adjusting Bitrate in `DeltaMask`. We ablate fundamental components in `DeltaMask`: the mechanism for sorting mask update indexes and our choice of probabilistic filter, assessing their impact on accuracy and bitrate. Using CLIP ViT-B/32 with N=10 under full participation, Fig.4a compares our entropy-based top_κ sorting with random sampling. The consistent performance gap highlights the importance of importance sampling. Increasing κ does not linearly enhance accuracy, peaking at κ=0.8. This suggests top_κ effectively filters noise by prioritizing updates with higher



(a) top$_\kappa$ percentages (b) Probabilistic filters

Figure 4. Impact of top_κ mechanism and probabilistic filter choice in `DeltaMask` performance in CIFAR-100 under IID settings.

certainty, reducing bitrate by transmitting less data. We also evaluate various probabilistic filters, focusing on bits-per-entry (*bpe*) from 8 to 32. Binary fuse filters (BFuse) generally outperform XoR filters in reducing bitrate without compromising accuracy, as shown in Fig.4b. Importantly, `DeltaMask` enables adjustable bitrate based on *bpe*, accommodating resource heterogeneity in FL.

4. Conclusions

We introduce `DeltaMask`, an FL technique for efficiently fine-tuning FMs under low bitrate constraints using stochastic masking and probabilistic filters for mask updates. Our evaluation shows `DeltaMask`'s effectiveness across various datasets and FMs, achieving significant communication reductions with performance comparable to traditional fine-tuning. Besides communication efficiency, `DeltaMask` can provide personalized FMs in FL, while it can be expanded to adapt a single FM to multiple tasks, each with its unique masks.

Acknowledgement

V.T's research is funded by the DAIS project, which has received funding from KDTJU under grant agreement No 101007273.

References

Aji, A. F. and Heafield, K. Sparse communication for distributed gradient descent. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2017. doi: 10.18653/v1/d17-1045. URL https://doi.org/10.18653%2Fv1%2Fd17-1045.

Alistarh, D., Grubic, D., Li, J., Tomioka, R., and Vojnovic, M. Qsgd: Communication-efficient sgd via gradient quantization and encoding, 2017.

Appleby, A. Murmurhash3. https://github.com/aappleby/smhasher/wiki/MurmurHash3, 2016. Accessed: 13/11/2023.

Chatterjee, S. and Diaconis, P. The sample size required in importance sampling, 2017.

Dehghani, M., Djolonga, J., Mustafa, B., Padlewski, P., Heek, J., Gilmer, J., Steiner, A., Caron, M., Geirhos, R., Alabdulmohsin, I., Jenatton, R., Beyer, L., Tschannen, M., Arnab, A., Wang, X., Riquelme, C., Minderer, M., Puigcerver, J., Evci, U., Kumar, M., van Steenkiste, S., Elsayed, G. F., Mahendran, A., Yu, F., Oliver, A., Huot, F., Bastings, J., Collier, M. P., Gritsenko, A., Birodkar, V., Vasconcelos, C., Tay, Y., Mensink, T., Kolesnikov, A., Pavetić, F., Tran, D., Kipf, T., Lučić, M., Zhai, X., Keysers, D., Harmsen, J., and Houlsby, N. Scaling vision transformers to 22 billion parameters, 2023.

Ferreira, P. A., da Silva, P. N., Gottin, V., Stelling, R., and Calmon, T. Bayesian signsgd optimizer for federated learning.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks, 2019.

Graf, T. M. and Lemire, D. Binary fuse filters: Fast and smaller than xor filters. *ACM J. Exp. Algorithmics*, 27, mar 2022. ISSN 1084-6654. doi: 10.1145/3510449. URL https://doi.org/10.1145/3510449.

Havasi, M., Peharz, R., and Hernández-Lobato, J. M. Minimal random code learning: Getting bits back from compressed model parameters, 2018.

Isik, B., Pase, F., Gunduz, D., Weissman, T., and Zorzi, M. Sparse random networks for communication-efficient federated learning, 2023.

Khalilian, S., Tsouvalas, V., Ozcelebi, T., and Meratnia, N. Fedcode: Communication-efficient federated learning via transferring codebooks, 2023.

Konečný, J., McMahan, H. B., Yu, F. X., Richtarik, P., Suresh, A. T., and Bacon, D. Federated learning: Strategies for improving communication efficiency. In *NIPS Workshop on Private Multi-Party Machine Learning*, 2016. URL https://arxiv.org/abs/1610.05492.

Kostopoulou, K., Xu, H., Dutta, A., Li, X., Ntoulas, A., and Kalnis, P. Deepreduce: A sparse-tensor communication framework for distributed deep learning, 2021.

Le, Y. and Yang, X. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

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, 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. SenSys '21, pp. 42–55, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450390972. doi: 10.1145/3485730.3485929. URL https://doi.org/10.1145/3485730.3485929.

Lin, Y., Han, S., Mao, H., Wang, Y., and Dally, W. J. Deep gradient compression: Reducing the communication bandwidth for distributed training, 2020.

Mallya, A., Davis, D., and Lazebnik, S. Piggyback: Adapting a single network to multiple tasks by learning to mask weights, 2018.

Mohtashami, A., Jaggi, M., and Stich, S. U. Masked training of neural networks with partial gradients, 2022.

Mozaffari, H., Shejwalkar, V., and Houmansadr, A. Frl: Federated rank learning, 2022.

Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., Assran, M., Ballas, N., Galuba, W., Howes, R., Huang, P.-Y., Li, S.-W., Misra, I., Rabbat, M., Sharma, V., Synnaeve, G., Xu, H., Jegou, H., Mairal, J., Labatut, P., Joulin, A., and Bojanowski, P. Dinov2: Learning robust visual features without supervision, 2023.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

Ramanujan, V., Wortsman, M., Kembhavi, A., Farhadi, A., and Rastegari, M. What's hidden in a randomly weighted neural network?, 2020.

Shysheya, A., Bronskill, J., Patacchiola, M., Nowozin, S., and Turner, R. E. Fit: Parameter efficient few-shot transfer learning for personalized and federated image classification. *arXiv preprint arXiv:2206.08671*, 2022.

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, 2022.

Vargaftik, S., Basat, R. B., Portnoy, A., Mendelson, G., Ben-Itzhak, Y., and Mitzenmacher, M. Drive: One-bit distributed mean estimation, 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 Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 21984–22014. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/vargaftik22a.html.

Wang, H., Sreenivasan, K., Rajput, S., Vishwakarma, H., Agarwal, S., Sohn, J.-y., Lee, K., and Papailiopoulos, D. Attack of the tails: Yes, you really can backdoor federated learning. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 16070–16084. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/b8ffa41d4e492f0fad2f13e29e1762eb-Paper.pdf.

Zhao, M., Lin, T., Mi, F., Jaggi, M., and Schütze, H. Masking as an efficient alternative to finetuning for pretrained language models. In Webber, B., Cohn, T., He, Y., and Liu, Y. (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2226–2241, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.174. URL https://aclanthology.org/2020.emnlp-main.174.

Zhou, H., Lan, J., Liu, R., and Yosinski, J. Deconstructing lottery tickets: Zeros, signs, and the supermask, 2020.

Zhuang, W., Chen, C., and Lyu, L. When foundation model meets federated learning: Motivations, challenges, and future directions. *arXiv preprint arXiv:2306.15546*, 2023.

A. `DeltaMask` Algorithm

We provide the pseudocode for `DeltaMask` in Algorithm 1. For completeness, we also provide the Bayesian Aggregation of compressed masks in Algorithm 2.

Algorithm 1 `DeltaMask`

Server initialize global model G with pretrained model weights w_{init}.
Server initialize mask weights $\theta^{\text{g},0} \in \mathbf{R}^d$ and Beta priors $\alpha^{g,0}$=$\beta^{g,0}$=λ_0.
for $r = 1, \ldots, R$ **do**
  Randomly select K clients to participate in round t
  for each client $k \in K$ **in parallel do**
    Sample binary server mask $m^{\text{g},t-1} \sim \text{Bern}_{t-1}(\theta^{\text{g},t-1})$
    $\theta^{k,t} \leftarrow \text{ClientUpdate}(\theta^{\text{g},t-1})$
    Sample binary mask $m^{\text{k},t} \sim \text{Bern}(\theta^{\text{k},t})$
    $\Delta'^{\text{k},t} \leftarrow \text{Sort}\ \{i \,|\, m^{\text{k},t} \neq m^{\text{g},t-1}\}_{i \in d}\ [1:\mathsf{K}]$ ▷ *// See Equation 4*
    $\mathcal{H}^{\text{k},t} \leftarrow \bigcup_{i \in \Delta'^{\text{k},t}} \phi(i)$ ▷ *// See Equation 1*
    $\text{PNG}^{k,t} \leftarrow \Psi(\mathcal{H}^{\text{k},t})$
  end for
  for each client $k \in K$ **do**
    $\mathcal{H}^{\text{k},t} \leftarrow \Psi^{-1}(\text{PNG}^{k,t})$
    $\hat{\Delta}'^{\text{k},t} \leftarrow \{i \mid \text{Member}\,(i) = \text{true}\}_{i \in d}$ ▷ *// See Equation 5*
    $m^{k,t} \leftarrow m^{g,t-1} \text{XOR}\ \mathcal{F}$ ▷ *// $\mathcal{F}$ is 1 in all positions of $\hat{\Delta}'^{\text{k},t}$ and 0 otherwise*
  end for
  $\theta^{g,t} \leftarrow \text{BayesAgg}(\{m^{k,t}\}_{k \in K}, t, \rho)$
end for

procedure ClientUpdate(θ)
  for epoch $e = 1, 2, \ldots, E$ **do**
    for batch $b \in \mathcal{D}^k$ **do**
      Sample a binary mask $m \sim \text{Bern}(\theta)$
      $\theta \leftarrow \theta - \eta \nabla_\theta \left(\mathcal{L}_{CE}\left(y, p_{m \odot w_{\text{init}}}\left(y|x_b\right)\right)\right)$
    end for
  end for
  return θ
end procedure

Algorithm 2 BayesAgg

Inputs: Clients' updates $\{m_{k,t}\}_{k \in K}$, federated round t, and client participation ρ.
Output: Global probability mask $\theta_{g,t}$.
if t % $(\frac{1}{\rho})$ = 0 **then**
  $\alpha_{g,t-1} = \beta_{g,t-1} = \lambda_0$
end if
$m^{\text{agg},t} \leftarrow \frac{1}{K} \sum_{k \in K} m^{k,t}$
$\alpha^{g,t} = \alpha^{g,t-1} + m^{\text{agg},t}$
$\beta^{g,t} = \beta^{g,t-1} + K \cdot 1 - m^{\text{agg},t}$
$\theta^{g,t} = \frac{\alpha^{g,t}}{\alpha^{g,t} + \beta^{g,t}}$
return $\theta^{g,t}$

B. Probabilistic Filters

Probabilistic filters are data structures that map a universe of keys, denoted as $\mathcal{U}$, of varying bit lengths, to fixed-size bit values, thereby compacting real-world data representations effectively. They achieve this by using hash functions to transform and store data in a uniformly distributed array, known as the fingerprints $\mathcal{H}$. This compact representation $\mathcal{H}$ facilitates efficient membership checking, with an adjustable rate of false positives — where a non-member might be incorrectly identified as a member — while ensuring zero false negatives. There are multiple variations of probabilistic filters, we focus on *binary fuse filters* (BFuse) (Graf & Lemire, 2022), which are known for their exceptional space efficiency and computational effectiveness. These filters offer a space efficiency of 8.62 bits per entry and a low false positive rate (up to 2^{-32}).

Formally, an μ-wise BFuse utilizes m distinct hash functions $h_j : \{0, 1, \ldots, 2^n - 1\} \rightarrow \{1, 2, \ldots, l\}$, for $j = 1, \ldots, \mu$, where l denotes the size of the fingerprints array, $\mathcal{H}$. Let $f : \mathbb{N} \rightarrow \{0, 1, \ldots, 2^n - 1\}$ be the fingerprint generation function, mapping each key to an n-bit value. For a given set of keys $\mathcal{U}$, we can compute the fingerprint array $\mathcal{H}$ as:

$$\mathcal{H} = \bigcup_{k \in \mathcal{U}} \phi(k) = \bigcup_{k \in \mathcal{U}} \left(\bigcup_{j=1}^{m} \{h_j(f(k))\} \right) \tag{3}$$

Here, $\phi(k)$ computes the set of m locations in $\mathcal{H}$ for each key k in $\mathcal{U}$. Once $\mathcal{H}$ is constructed, we can perform a membership check as:

$$\text{Member}(x) = \begin{cases} \text{true}, & \bigoplus_{j=1}^{m} \mathcal{H}\left[h_j(f(x))\right] = f(x) \\ \text{false}, & \text{otherwise} \end{cases} \tag{4}$$

where, $\bigoplus_{j=1}^{m} \mathcal{H}[\cdot]$ represents the bitwise XOR operation performed on the array values of $\mathcal{H}$, indicated by the hash functions $h_j(f(x))$. The *Member*$(\cdot)$ function returns true if the result of the XOR operation over $\mathcal{H}$ matches the fingerprint $f(x)$, suggesting that x is likely a member of the set, and false in all other occasions. Note that while computing a large number of hashes may seem daunting, not all hashing algorithms are computationally intensive. For example, BFuse use MurmurHash3 (Appleby, 2016), which is computationally efficient and exhibits exceptional properties for hashing large data structures into space-efficient arrays (e.g., uniform hash distribution and randomness).

C. Stochastic Mask Training

Unlike the thresholding mechanisms (Li et al., 2021; Vallapuram et al., 2022; Mozaffari et al., 2022) that creates binary masks by clipping mask scores $s \in \mathbb{R}^d$, stochastic mask training (Isik et al., 2023), involves drawing a binary mask $m \in \{0, 1\}^d$ from the underlying mask's probability θ using the Bernoulli distribution (noted as $m \sim \text{Bern}(\theta)$). To generate θ from the *unbounded* mask scores s, a sigmoid transformation is applied (i.e., $\theta = \text{Sigmoid}(s)$). Hence, m is used during the forward pass to compute the loss $\mathcal{L}(\cdot)$, and θ is subsequently updated through back-propagation. As the values of s remain *unbounded*, it allows for an unbiased estimation of the true aggregate of the local clients' mask probabilities through Bayesian aggregation (Ferreira et al.). Specifically, it refines the global model at round t in federated setting by treating the stochastic mask's probability $\theta^{\mathrm{g},t}$ as a Beta distribution $\text{Beta}(\alpha_{\mathrm{g},t}, \beta_{\mathrm{g},t})$, with $\alpha_{\mathrm{g},t}$ and $\beta_{\mathrm{g},t}$ initialized to λ_0. These parameters are updated with the aggregated local binary masks from participating clients (denoted as $\bar{\theta}^{\mathrm{g},t}$), computed as $\alpha_{\mathrm{g},t} = \alpha_{\mathrm{g},t-1} + \bar{\theta}^{\mathrm{g},t}$ and $\beta_{g,t} = \beta_{g,t-1} + K \cdot \mathbf{1}_d - \bar{\theta}^{\mathrm{g},t}$. The aggregated probability mask is then calculated by:

$$\theta^{g,t} = \frac{\alpha_{g,t} - 1}{\alpha_{g,t} + \beta_{g,t} - 2}, \tag{5}$$

where the division is performed element-wise division. For best performance, α and β are periodically reset to $\lambda_0 = 1$ at a rate inverse to the participation rate p (Isik et al., 2023). It's important to note that while the model's weight values remain unchanged, the binary mask m selectively activates neurons by element-wise multiplication with the initialized model weights w_{init}, denoted as $\dot{w_{\mathrm{k},t}} = m^{\mathrm{k},t} \odot w_{\text{init}}$.

D. Distributed Mean Estimation Error Analysis

We now provide proof of the upper bound on the estimation error of `DeltaMask`. Recall that we use probabilistic filters to reconstruct clients' binary masks, $m^{k,t} \sim \text{Bern}(\theta^{k,t})$ on server-side, which introduce an independent (across both clients and mask dimensions) "bit-flip" error probability 2^{-p} (p referring to the false positive rate of the filter). We refer to these reconstructed masks as $m'^{k,t}$. Here, our true mean is $\bar{\theta}^{g,t} = \frac{1}{K} \sum_{k \in K_t} \theta_k^t$, while our estimate is $\hat{\bar{\theta}}^{g,t} = \frac{1}{K} \sum_{k \in K_t} m'^{k,t}$. Furthermore, we use capital letters to refer to random variables, while small letters refer to their deterministic quantities. We can then compute the error as follows:

$$\mathbb{E}_{M^{k,t}\sim\text{Bern}(\theta^{k,t})\forall k\in K}\left[\left\|\bar{\theta}^{g,t}-\hat{\bar{\theta}}^{g,t}\right\|_2^2\right]=\sum_{i=1}^{d}\mathbb{E}_{M_i^{k,t}\sim\text{Bern}(\theta_i^{k,t})\forall k\in K}\left[\left(\bar{\theta}^{g,t}-\hat{\bar{\theta}}^{k,t}\right)^2\right] \tag{6}$$

$$=\sum_{i=1}^{d}\mathbb{E}_{M_i^{k,t}\sim\text{Bern}(\theta_i^{k,t})\forall k\in K}\left[\left(\frac{1}{K}\sum_{k\in K}\left(M'^{k,t}_i-\theta_i^{k,t}\right)\right)^2\right] \tag{7}$$

$$=\frac{1}{K^2}\sum_{i=1}^{d}\mathbb{E}_{M_i^{k,t}\sim\text{Bern}(\theta_i^{k,t})\forall k\in K}\left[\left(\sum_{k\in K}\left(M'^{k,t}_i-\theta_i^{k,t}\right)\right)^2\right] \tag{8}$$

$$=\frac{1}{K^2}\sum_{i=1}^{d}\sum_{k\in K}\mathbb{E}_{M_i^{k,t}\sim\text{Bern}(\theta_i^{k,t})}\left[\left(M'^{k,t}_i-\theta_i^{k,t}\right)^2\right] \tag{9}$$

$$=\frac{1}{K^2}\sum_{i=1}^{d}\sum_{k\in K}\Big(\mathbb{E}_{M_i^{k,t}\sim\text{Bern}(\theta_i^{k,t})}\left[(M'^{k,t}_i)^2\right] -2\theta_i^{k,t}\mathbb{E}_{M_i^{k,t}\sim\text{Bern}(\theta_i^{k,t})}\left[M'^{k,t}_i\right]+(\theta_i^{k,t})^2\Big) \tag{10}$$

$$=\frac{1}{K^2}\sum_{i=1}^{d}\sum_{k\in K}(\theta_i^{k,t}-(\theta_i^{k,t})^2)-4\cdot(2^{-p})(\theta_i^{k,t}-(\theta_i^{k,t})^2)+2^{-p} \tag{11}$$

$$\leq\frac{d}{4K} \tag{12}$$

We begin by expressing the expected squared L^2 norm of the error $\hat{\bar{\theta}}^{g,t}$ and $\bar{\theta}^{g,t}$ in (7). From (7) to (8), we use the definitions of $\hat{\bar{\theta}}^{g,t}=\frac{1}{K}\sum_{k\in K_t}m'^{k,t}$ and $\bar{\theta}^{g,t}=\frac{1}{K}\sum_{k\in K_t}\theta_k^t$. To move from (9) to (10), we use the fact that $M'^{k,t}$ and $M'^{l,t}$ are independent for $k\neq l$; thus the expected value of cross-product terms in the expansion of squared sum of is zero. At (11), we utilize the fact that $M'^{k,t}_i$ is a Bernoulli variable (meaning $(M'^{k,t}_i)^2=M'^{k,t}_i$) and introduce the "*bit-flip*" error probability 2^{-p} due to the probabilistic filters; thus its expected value is $E[(M'^{k,t}_i]=(1-2^{-p})\theta_i^{k,t}+2^{-p}\theta_i^{k,t}$. Finally, in (13), given that the variance of a Bernoulli random variable is maximized when the probability of success is 0.5, and that the flipping process does not change this maximum possible variance – as $2^{-p}<<1$ given $p\in\{8,16,32\}$ –we concluded that the upper bound of the expected squared error is $\frac{d}{4K}$, where d is the number of dimensions and K is the number of clients. It is important to note that our probabilistic filter-based encoding provides the same upper-bound estimation error as (Isik et al., 2023); yet, it can achieve significant reductions in terms of required bpp for transmitting masks.

E. Privacy Implications of DeltaMask

In FL, protecting privacy is essential, as model updates might inadvertently expose client data (Wang et al., 2020). `DeltaMask` employs probabilistic filters like binary fuse filters for stochastic mask updates, enhancing data privacy due to their reliance on multiple hashing operations sensitive to initial conditions. Establishing an initial *seed* through a secure channel with the server — leveraging a public-private key pairing — we mitigate the risk of eavesdropping on client updates to performing model inversion attacks. Crucially, the probabilistic filters' false positive rate, akin to an independent *bit-flipping* probability, functions as a local differential privacy safeguard. While providing absolute privacy guarantees is not the primary objective of `DeltaMask`, its hashing operations inherently boost privacy, a beneficial side effect. We leave further exploration of this to future work.

F. Additional Experiments

F.1. Additional Experimental Details

Training Parameters: For our experiments, clients completed 1 local epoch per round with a batch size of 64 and Adam optimizer with a learning rate of 0.1. We adopted Bayesian aggregation, resetting the prior every $\frac{1}{\rho}$ rounds, where ρ is the participation rate (as per (Isik et al., 2023)). In scenarios where ρ is less than 1.0, client selection in each round was randomized. In most experiments, we set κ to 0.8, except for those detailed in Fig.4a. We conducted 100 federated rounds for experiments with ρ=1 (both IID and non-IID settings) and increased the number of rounds to 200 and 300 for IID and non-IID experiments, respectively, when $\rho<<1$. Unless otherwise mentioned, we employed CLIP ViT-B/32 for experiments involving CLIP. We perform 3 independent runs and report the average accuracy on test-set in all our experiments.

Weight Initialization: The neural network $p_{w_{\text{init}}}$ is initialized using weights $w_{\text{init}}=(w_{\text{init},1},w_{\text{init},2},\dots,w_{\text{init},d})\in\mathbb{R}^d$ derived from a pre-trained foundation model, yet, the classification head for downstream tasks is randomly initialized. This means that while the pre-trained backbone offers high-quality features useful across various tasks, the randomly initialized classifier head significantly influences the model's overall performance. Prior research has sampled weights from a uniform distribution around the Kaiming initialization to find highly-performing subnetworks on randomly initialized network (Isik et al., 2023; Zhou et al., 2020; Ramanujan et al., 2020; Zhou et al., 2020). However, as we focus on pre-trained models, we allow the classification head to adapt during a single round of linear probing, where the rest of the model remains frozen. This yields more stable results and rapid convergence. For a fair comparison, we employ identical weights initialization methods across all considered baselines. We also investigate scenarios with extremely low bitrates, where, linear probing is not feasible in Appx. F.7.

Baselines Configuration: For FedMask, we set a binary threshold τ (masking with m_i=1 if $\theta_i \geq \tau$, and 0 otherwise) in the range [0.4, 0.6] for IID and [0.2, 0.0] for non-IID experiments, aligning with (Isik et al., 2023). In EDEN, a 1-bit gradient compression scheme was used to match the bitrate (*bpp*) of other baselines. Notably, EDEN's compression is model-dependent but yields nearly constant *bpp* reductions across all experiments. From DeepReduce compression, we discard the values' compression stage (as we deal we binary masks), and utilize only the Bloom filter-based index compression using the $P0$-policy (Kostopoulou et al., 2021). Here, binary masks were learned via stochastic mask training ((Isik et al., 2023)), ensuring operation near the 1 *bpp* regime and facilitating clear comparison with `DeltaMask`. For our comparison with FedPM, we use identical settings albeit the compression scheme with probabilistic filters of `DeltaMask`, to clearly illustrate the benefits of our approach. We conducted our experiments on NVIDIA A10 GPUs on an internal cluster server, using 2 GPUs per one run.

F.2. Additional Experiments in IID settings

In this section, we present additional experiments conducted under IID settings with varying participation rates (ρ). To ensure a fair comparison, we included both Linear Probing, which involves adapting a single linear classifier atop the (frozen) pre-trained model, and full Fine-tuning, wherein only the layers modified in `DeltaMask` are fine-tuned. In Table 1, apart from report models' accuracies across tasks, we include the average *bpp* and accuracy across all tasks for a concise comparison.

Table 1. Performance evaluation of **`DeltaMask` (Ours)** in terms of average bitrate (bits-per-parameter) during FL training using $Dir(10)$ over classes ($C_p \approx 1.0$ / **IID settings**) for CLIP ViT-B/32. Federated parameters are set to N=30 and E=1. For $\rho < 1$, clients are randomly selected.

	Method	CIFAR-10	CIFAR-100	SVHN	EMNIST	Fashion-MNIST	EuroSAT	Food-101	Cars196	Avg. Acc	Avg. bpp
$\rho = 0.2$	Linear Probing	92.12 ± 0.007	67.23 ± 0.011	59.70 ± 0.016	89.89 ± 0.008	89.05 ± 0.010	94.81 ± 0.009	67.58 ± 0.014	59.87 ± 0.016	77.51	-
	Fine-tuning	94.38 ± 0.013	76.12 ± 0.019	91.88 ± 0.012	94.02 ± 0.018	92.54 ± 0.009	97.61 ± 0.015	85.73 ± 0.017	66.98 ± 0.011	**87.48**	32
	FedMask	85.32 ± 0.033	61.38 ± 0.057	68.71 ± 0.046	81.32 ± 0.024	84.32 ± 0.044	92.01 ± 0.025	62.28 ± 0.037	57.12 ± 0.029	74.06	1.0
	EDEN	87.11 ± 0.006	65.89 ± 0.009	79.16 ± 0.008	86.36 ± 0.006	85.21 ± 0.012	91.24 ± 0.010	69.59 ± 0.012	62.07 ± 0.011	78.33	0.703
	DeepReduce	86.71 ± 0.071	64.98 ± 0.091	60.32 ± 0.061	84.42 ± 0.044	84.09 ± 0.057	92.37 ± 0.041	64.91 ± 0.043	55.72 ± 0.078	73.61	1.123
	FedPM	90.31 ± 0.016	74.66 ± 0.019	87.03 ± 0.017	91.42 ± 0.021	89.79 ± 0.013	95.57 ± 0.015	74.80 ± 0.014	62.19 ± 0.017	83.22	0.946
	`DeltaMask`	89.52 ± 0.021	74.01 ± 0.033	86.86 ± 0.024	92.27 ± 0.027	89.68 ± 0.014	94.94 ± 0.019	74.09 ± 0.029	61.56 ± 0.030	82.87	**0.197**
$\rho = 1.0$	Linear Probing	93.97 ± 0.004	74.11 ± 0.009	59.26 ± 0.011	89.40 ± 0.008	89.47 ± 0.005	95.35 ± 0.003	76.64 ± 0.009	61.72 ± 0.012	79.99	-
	Fine-tuning	94.50 ± 0.010	77.35 ± 0.009	92.72 ± 0.012	94.89 ± 0.010	92.98 ± 0.013	98.24 ± 0.011	86.72 ± 0.009	67.23 ± 0.014	**88.08**	32
	FedMask	90.84 ± 0.028	70.64 ± 0.057	74.32 ± 0.039	84.22 ± 0.031	88.64 ± 0.029	95.09 ± 0.038	68.46 ± 0.034	61.59 ± 0.039	79.23	1.0
	EDEN	93.15 ± 0.009	72.02 ± 0.010	86.67 ± 0.007	91.55 ± 0.009	90.40 ± 0.012	95.34 ± 0.010	80.02 ± 0.004	63.98 ± 0.008	84.14	0.691
	DeepReduce	88.17 ± 0.034	68.59 ± 0.069	62.34 ± 0.056	86.92 ± 0.073	85.44 ± 0.031	94.12 ± 0.043	67.92 ± 0.075	58.42 ± 0.041	76.53	1.089
	FedPM	93.58 ± 0.014	75.56 ± 0.011	88.76 ± 0.013	93.45 ± 0.015	92.10 ± 0.009	96.45 ± 0.019	83.45 ± 0.013	65.23 ± 0.014	86.07	0.872
	`DeltaMask`	93.50 ± 0.019	74.82 ± 0.023	87.95 ± 0.021	92.52 ± 0.019	91.27 ± 0.023	95.64 ± 0.017	82.73 ± 0.024	64.94 ± 0.026	85.44	**0.151**

In Table 1, we note that `DeltaMask` achieves significant reductions in bitrate, while maintaining performance on par with fine-tuning. This is particularly evident in scenarios with ρ less than 1, where `DeltaMask` ability to reduce bitrate without compromising on accuracy highlights its effectiveness in federated learning environments with varying levels of client participation.

F.3. Additional Experiments in non-IID settings

In this section, we provide additional experiments performed under non-IID settings, where we varied the participation rate (ρ). Similar to F.2, we include both Linear Probing and Fine-tuning for a rigorous evaluation. We report our findings in Table 2, where we also report the average *bpp* and accuracy across all tasks for a concise comparison of our baselines.

Table 2 reveals a notable improvement in `DeltaMask` performance, especially when the participation ratio ρ is less than 1, with only a 2% accuracy difference compared to fine-tuning. This is a critical observation, since non-IID data distributions coupled with partial client participation closely mirror the conditions of real-world federated settings. Furthermore, our analysis shows that methods using stochastic mask training, such as DeepReduce and FedPM, yield better final model accuracy under non-IID conditions than traditional compression schemes like EDEN or hard-thresholding masking techniques like FedMask. Interestingly, the CLIP ViT-B/32 model excels in non-IID scenarios, underscoring the robust generalization abilities of pre-trained foundation models, which are particularly advantageous in non-IID federated environments. This emphasizes the importance of adapting these models for edge computing, capitalizing on their capability to effectively handle diverse and complex data distributions.

Table 2. Performance evaluation of **DeltaMask (Ours)** in terms of average bitrate (bits-per-parameter) during FL training using $Dir(0.1)$ over classes ($C_p \approx 0.2$ / **non-IID settings**) for CLIP ViT-B/32. Federated parameters are set to N=30 and E=1. For $\rho < 1$, clients are randomly selected.

	Method	CIFAR-10	CIFAR-100	SVHN	EMNIST	Fashion-MNIST	EuroSAT	Food-101	Cars196	Avg. Acc	Avg. bpp
$\rho = 0.2$	Linear Probing	84.51 ± 0.019	49.04 ± 0.022	43.16 ± 0.020	82.41 ± 0.035	86.29 ± 0.024	91.63 ± 0.022	51.54 ± 0.021	47.92 ± 0.038	67.06	-
	Fine-tuning	92.59 ± 0.024	70.20 ± 0.037	87.39 ± 0.036	92.00 ± 0.057	88.25 ± 0.039	95.56 ± 0.029	79.38 ± 0.034	60.11 ± 0.051	**83.19**	32
	FedMask	83.14 ± 0.059	51.66 ± 0.119	51.78 ± 0.049	83.75 ± 0.078	85.91 ± 0.073	90.05 ± 0.074	53.19 ± 0.063	51.37 ± 0.105	68.86	1.0
	EDEN	87.87 ± 0.037	64.62 ± 0.106	81.06 ± 0.081	86.73 ± 0.050	86.75 ± 0.055	90.22 ± 0.062	72.55 ± 0.056	58.71 ± 0.034	78.56	0.715
	DeepReduce	86.07 ± 0.097	64.39 ± 0.088	82.92 ± 0.071	85.14 ± 0.084	83.91 ± 0.067	86.12 ± 0.117	52.92 ± 0.055	49.72 ± 0.110	73.90	1.173
	FedPM	90.70 ± 0.045	67.42 ± 0.095	87.51 ± 0.079	89.77 ± 0.095	88.42 ± 0.092	93.57 ± 0.067	76.80 ± 0.076	59.06 ± 0.098	81.64	0.948
	DeltaMask	90.32 ± 0.083	66.90 ± 0.051	87.36 ± 0.093	89.09 ± 0.047	86.91 ± 0.067	93.54 ± 0.101	76.39 ± 0.086	58.52 ± 0.102	81.13	**0.233**
$\rho = 1.0$	Linear Probing	91.46 ± 0.026	71.96 ± 0.025	46.03 ± 0.017	84.57 ± 0.031	87.13 ± 0.018	92.98 ± 0.016	68.70 ± 0.028	54.03 ± 0.032	74.61	-
	Fine-tuning	93.61 ± 0.048	75.49 ± 0.052	90.10 ± 0.063	93.13 ± 0.037	91.06 ± 0.041	97.02 ± 0.034	84.71 ± 0.013	64.93 ± 0.063	**86.26**	32
	FedMask	88.42 ± 0.051	63.04 ± 0.081	64.32 ± 0.073	86.41 ± 0.039	86.39 ± 0.044	91.67 ± 0.031	68.04 ± 0.050	54.39 ± 0.089	75.34	1.0
	EDEN	92.14 ± 0.043	71.65 ± 0.060	86.28 ± 0.057	90.87 ± 0.046	89.94 ± 0.034	93.26 ± 0.035	78.79 ± 0.083	61.18 ± 0.027	83.01	0.703
	DeepReduce	87.33 ± 0.052	67.19 ± 0.061	83.19 ± 0.048	85.71 ± 0.082	84.52 ± 0.075	92.12 ± 0.060	69.11 ± 0.092	60.31 ± 0.094	78.69	1.092
	FedPM	92.99 ± 0.045	74.34 ± 0.023	89.35 ± 0.025	92.65 ± 0.098	91.33 ± 0.041	95.37 ± 0.048	83.69 ± 0.076	63.65 ± 0.074	85.42	0.901
	DeltaMask	92.84 ± 0.083	73.69 ± 0.051	89.01 ± 0.085	91.92 ± 0.089	91.27 ± 0.055	94.54 ± 0.103	83.48 ± 0.081	63.47 ± 0.096	85.03	**0.191**

F.4. Experiments in ImageNet datasets

In this section, we extend our evaluation in more complex tasks to assess the effectiveness of DeltaMask to fine-tune FMs in federated settings in a communication-efficient manner under datasets of larger complexity. For this, we perform experiments on Tiny-ImageNet (Le & Yang, 2015) with both CLIP ViT-B/32 and CLIP ViT-L/14. The results, reported on Table 3 showcase that DeltaMask can effectively fine-tune FMs in more complex tasks, such as ImageNet datasets, while maintaining the same efficiency in terms of *bpp*.

Table 3. Evaluation of DeltaMask using $Dir(10.0)$ over classes ($C_p \approx 1.0$ / IID settings) across CLIP architectures in Tiny-ImageNet (Le & Yang, 2015).

Method	CLIP ViT-B/32		CLIP ViT-L/14	
	Accuracy	Avg. bpp	Accuracy	Avg. bpp
Fine-tuning	86.12	32	89.02	32
FedPM	84.22	0.871	87.04	0.862
DeltaMask (Ours)	83.76	**0.201**	86.57	**0.218**

F.5. Generalization across Neural Architectures and Pre-training Strategies

Here, we evaluate DeltaMask ability to work across various neural architectures pre-trained in a different self-supervised manner. We train masks for downstream task adaptation in a communication-constrained FL environment. For this, we perform experiments with N=10 on additional (larger) ViT architectures, namely CLIP-Large and DINOv2-Large, as well as a pure convolution-based architecture, ConvMixer-768/32 on CIFAR-100 as a downstream classification task. In all experiments, we mask the last 5 blocks, as discussed in Sec. 3. From Table 4, DeltaMask demonstrates robust adaptability across diverse pre-trained architectures in a FL setup with communication constraints. Notably, DeltaMask performance on large ViT architectures yield accuracies near those of fine-tuning, notably with CLIP ViT-L/14 slightly surpassing it. This is significant, considering the communication efficiency depicted by the average bitrate, which remains close to 0.2 bpp across all architectures. ConvMixer-768/32 also adapts well with DeltaMask, showing a modest accuracy reduction while meeting communication constraints. These results reinforce our method's suitability across diverse architectures, allowing for communication-efficient downstream task adaptation of FMs in a federated setting.

Table 4. Evaluation of DeltaMask using $Dir(10.0)$ over classes ($C_p \approx 1.0$ / IID settings) across architectures and pre-training strategies. Federated parameters are set to N=10, ρ=1 and E=1.

Metric	CLIP ViT-B/32	CLIP ViT-L/14	DINOv2-Base	DINOv2-Small	ConvMixer-768/32
Fine-tuning	77.35 ± 0.009	89.07 ± 0.012	75.01 ± 0.007	65.55 ± 0.019	78.52 ± 0.009
DeltaMask (Ours)	75.82 ± 0.023	89.48 ± 0.031	73.36 ± 0.027	63.01 ± 0.033	75.31 ± 0.021
Avg. bpp	0.207 ± 0.001	0.225 ± 0.002	0.197 ± 0.001	0.214 ± 0.001	0.251 ± 0.001

F.6. `DeltaMask` Efficiency on Edge Devices

In this section, we evaluate the runtime resource demands—computation and energy—of our probabilistic filter compression on three popular embedded platforms: NVIDIA Jetson Nano (4GB), Raspberry Pi 4 (4GB), and Coral Dev Board (1GB). These platforms were selected for their widespread use and capability to run machine learning tasks at the edge. To measure energy consumption, we used a Raspberry Pi 4 equipped with a Current/Power Monitor HAT, monitoring each device's energy use with a 0.1 Ohm sampling resistor. Our tests, conducted over 5 runs, record the average runtime (in milliseconds) and energy usage (in nano Joules) for different probabilistic filters with varying bits per entry (8, 16, and 32), as detailed in Table 5.

Table 5. Average energy and latency benchmarking of the considered probabilistic filters across different devices. The CPU execution time (ms) and estimated energy consumption (nJ) per entry is computed over 10M entries.

Filter	Metric	Raspberry Pi 4	Coral Dev Board	Jetson Nano
`Xor8`	CPU Execution Time (ms)	0.942 ± 0.0165	1.682 ± 0.0059	0.479 ± 0.0001
	Energy Consumption (nJ)	3.223 ± 0.0023	2.826 ± 0.0011	2.334 ± 0.0012
`Xor16`	CPU Execution Time (ms)	0.955 ± 0.0250	1.683 ± 0.0008	0.502 ± 0.0001
	Energy Consumption (nJ)	4.052 ± 0.0032	3.580 ± 0.0003	3.386 ± 0.0016
`Xor32`	CPU Execution Time (ms)	0.978 ± 0.0278	1.701 ± 0.0006	0.539 ± 0.0005
	Energy Consumption (nJ)	6.292 ± 0.0021	4.732 ± 0.0008	4.692 ± 0.0023
`BFuse8`	CPU Execution Time (ms)	**0.587 ± 0.0059**	**1.144 ± 0.0035**	**0.289 ± 0.0013**
	Energy Consumption (nJ)	**2.045 ± 0.0019**	**1.979 ± 0.0015**	**1.829 ± 0.0023**
`BFuse16`	CPU Execution Time (ms)	0.590 ± 0.0066	1.183 ± 0.0029	0.282 ± 0.0002
	Energy Consumption (nJ)	3.262 ± 0.0020	2.898 ± 0.0017	2.157 ± 0.0033
`BFuse32`	CPU Execution Time (ms)	0.612 ± 0.0054	1.201 ± 0.0017	0.301 ± 0.0002
	Energy Consumption (nJ)	4.021 ± 0.0026	3.771 ± 0.0022	3.263 ± 0.0012

From the results, we clearly notice that all filter variants demand limited computational resources, both in terms of execution time and energy requirements. `BFuse8` is particularly notable for its efficiency, requiring only an average execution time of 0.673 milliseconds and consuming just 1.95 nano Joules of energy across the considered devices. This underscores the practicality of our probabilistic filter-based compression scheme in federated settings, where devices are often constrained by limited computational capabilities and strict energy budgets. Additionally, our analysis shows that even with an increase in the bits-per-entry (*bpe*) parameter, the rise in execution time and energy consumption is quite modest. This is particularly noteworthy given the simultaneous improvement in the false positive rate, which is inversely proportional to $2^{-\text{bpe}}$. This pattern suggests a beneficial trade-off between accuracy and resource utilization, reinforcing the adaptability and effectiveness of our approach in federated learning scenarios that prioritize computational efficiency and energy conservation.

F.7. Comparing Classifier Heads in `DeltaMask`

In `DeltaMask`, we enable the classification head to adapt in a single linear probing round, while freezing the rest of the model. This approach produces more stable outcomes and quicker convergence than previous methods (Isik et al., 2023; Zhou et al., 2020; Ramanujan et al., 2020; Zhou et al., 2020) that used Kaiming initialization to identify high-performing subnetworks in randomly initialized networks. Although the classification head typically has fewer parameters, scenarios requiring extremely low bitrates make transmitting even a single round's floating-point weights impractical. In this section, we explore such situations, investigating different alternatives for the classifier layer. Specifically, we replace the linear classifier with a Gaussian Naive Bayes classifier from FiT (Shysheya et al., 2022), specifically *FIT-LDA*. This classifier is data-driven, with a minimal number of learnable parameters (2 float-point values), making it ideal for our purpose. In our analysis, we utilize CLIP ViT-B/32, masking the last five transformer blocks and compare $\texttt{DeltaMask}_{\text{FiT}}$ against both a single-round trained linear classifier ($\texttt{DeltaMask}_{\text{LP}}$) and a Kaiming initialized (frozen) classifier ($\texttt{DeltaMask}_{\text{He}}$).

Table 6. Evaluating Classifier Initialization Schemes in **`DeltaMask`**. Comparing Average Bitrate and Accuracy in FL Training using $Dir(10)$ over classes ($C_p{\approx}1.0$ / **IID settings**) for CLIP ViT-B/32. Federated parameters are set to N=30 and E=1.

Method	CIFAR-10	CIFAR-100	SVHN	EMNIST	Fashion-MNIST	EuroSAT	Food-101	Cars196	Avg. Acc	Avg. bpp
Fine-tuning	94.50 ± 0.010	77.35 ± 0.009	92.72 ± 0.012	94.89 ± 0.010	92.98 ± 0.013	98.24 ± 0.011	86.72 ± 0.009	67.23 ± 0.014	88.08	32
$\texttt{DeltaMask}_{\text{He}}$	90.28 ± 0.052	67.34 ± 0.069	84.09 ± 0.063	87.32 ± 0.081	87.69 ± 0.034	93.22 ± 0.073	78.05 ± 0.028	58.74 ± 0.084	80.84	**0.143**
$\texttt{DeltaMask}_{\text{FiT}}$	93.42 ± 0.023	71.17 ± 0.041	86.31 ± 0.039	92.09 ± 0.021	89.87 ± 0.026	95.53 ± 0.019	81.71 ± 0.033	60.01 ± 0.029	83.76	0.145
$\texttt{DeltaMask}_{\text{LP}}$	93.50 ± 0.019	74.82 ± 0.023	87.95 ± 0.021	92.52 ± 0.019	91.27 ± 0.023	95.64 ± 0.017	82.73 ± 0.024	64.94 ± 0.026	**85.44**	0.151

From Table 6, we notice that $\texttt{DeltaMask}_{\text{LP}}$ outperforms other initialization methods by over 2% without significantly increasing the bitrate, while *FiT* can be an effective alternative to Kaiming initialization, increasing accuracy by $\approx$3%. More importantly, these findings highlight the importance of appropriate classifier layer initialization during fine-tuning of foundation models in downstream tasks. However, we demonstrate that a single fine-tuning round of the classifier layer, with the remaining model frozen, is an effective strategy with minimal impact on the communicated bitrate.