FedSZ: Leveraging Error-Bounded Lossy Compression for Federated Learning Communications

Grant Wilkins*†, Sheng Di* Jon C. Calhoun‡, Zilinghan Li*§, Kibaek Kim*, Robert Underwood*,
Richard Mortier†, Franck Cappello*
*Argonne National Laboratory, Lemont, IL, USA
†University of Cambridge, Cambridge, UK
‡Clemson University, Clemson, SC, USA
§University of Illinois at Urbana-Champaign, Urbana, IL, USA
Emails: {gfw27,rmm1002}@cam.ac.uk, {sdi1,kimk,runderwood,cappello}@anl.gov, jonccal@clemson.edu, zl52@illinois.edu

2024 IEEE 44th International Conference on Distributed Computing Systems (ICDCS) | 979-8-3503-8605-9/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICDCS60910.2024.00060

***Abstract*—With the promise of federated learning (FL) to allow for geographically-distributed and highly personalized services, the efficient exchange of model updates between clients and servers becomes crucial. FL, though decentralized, often faces communication bottlenecks, especially in resource-constrained scenarios. Existing data compression techniques like gradient sparsification, quantization, and pruning offer some solutions, but may compromise model performance or necessitate expensive retraining. In this paper, we introduce FEDSZ, a specialized lossy-compression algorithm designed to minimize the size of client model updates in FL. FEDSZ incorporates a comprehensive compression pipeline featuring data partitioning, lossy and lossless compression of model parameters and metadata, and serialization. We evaluate FEDSZ using a suite of error-bounded lossy compressors, ultimately finding `SZ2` to be the most effective across various model architectures and datasets including AlexNet, MobileNetV2, ResNet50, CIFAR-10, Caltech101, and Fashion-MNIST. Our study reveals that a relative error bound 10^{-2} achieves an optimal tradeoff, compressing model states between 5.55-$12.61\times$ while maintaining inference accuracy within $< 0.5\%$ of uncompressed results. Additionally, the runtime overhead of FEDSZ is $< 4.7\%$ or between of the wall-clock communication-round time, a worthwhile trade-off for reducing network transfer times by an order of magnitude for networks bandwidths $< 350\text{Mbps}$. Intriguingly, we also find that the error introduced by FEDSZ could potentially serve as a source of differentially private noise, opening up new avenues for privacy-preserving FL.**

***Index Terms*—Lossy compression, federated learning, edge computing**

I. INTRODUCTION

Federated learning (FL), a decentralized machine learning (ML) paradigm, has found applications in a plethora of fields, including image classification and generative text models on mobile phones, real-time decision-making on edge devices, large-scale anomaly detection, and personalization in healthcare [1]. However, FL's broad adoption and effectiveness face challenges due to the high complexity of models, substantial computational demands and ever-increasing scale. Modern FL systems often require processing and training models, each with billions of parameters, while communicating with thousands of distributed clients, bringing a heavy computational and communication overhead.

In FL, client-server communication can become a significant bottleneck to achieving scalability of number of clients and server-side robustness [2]–[4]. For instance, an autonomous vehicle can generate up to 1 GB/s of sensor or image data for on-device training and validation [5]. If a 10GB client update is sent to a server via a mobile network with a bandwidth of 10Mbps, it would take approximately 150 minutes to transmit. With several other computational tasks happening and an autonomous vehicle being battery-constrained, this is a significant amount of effort spent on communication alone. As a result, strategies to cut data-related communication challenges are necessary to enhance the scalability and robustness of FL [6].

Error-bounded lossy compression (EBLC) [7]–[9] is widely used to significantly reduce large volumes of data generated by high-performance computing (HPC) simulations, but how EBLC can be used to mitigate the communication cost for FL significantly is still an open question. Moreover, introducing EBLC produces many new challenges for FL: (1) The FL environment fundamentally differs from HPC. In the area of HPC, the compute or storage nodes are generally under centralized management in a supercomputer, where the related resources are relatively stable. However, in FL, the distributed clients could be any device (e.g., Raspberry Pi) on a wide area network (WAN), projecting a fairly non-deterministic environment with diverse communication bandwidths and node compute power. So, a lossy compression method has to consider data fidelity, network latency, and device processing power. (2) Scientific datasets are generally much more compressible than the parameter datasets generated by FL clients. The key reason is that the scientific datasets generally correspond to simulating or capturing a physical, chemical or biological phenomenon. In comparison, the FL parameter data are often irregular/spiky, to be shown in Section V-A.

In this paper, we address several key questions to confirm our hypothesis that *EBLC significantly cuts client-server communication runtime while preserving FL model accuracy.*

Corresponding author: Sheng Di, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 Cass Avenue, Lemont, IL 60439, USA

2575-8411/24/$31.00 ©2024 IEEE
DOI 10.1109/ICDCS60910.2024.00060

Authorized licensed use limited to: CLEMSON UNIVERSITY. Downloaded on September 19,2024 at 02:45:08 UTC from IEEE Xplore. Restrictions apply.

1) How can we integrate EBLC into the FL communication pipeline?
2) Given the uniqueness and complexity of model parameters in FL, which EBLC algorithm yields the best trade-off between size reduction and retaining model data fidelity?
3) Lossy compression, while reducing data size, introduces both computational overhead and data error. Is the runtime and error cost incurred by these factors outweighed by the benefits of reduced communication time?

In this study, we present a novel optimized, practical EBLC heuristic, FEDSZ, designed to compress FL client model updates and significantly reduce client-server communication costs. In particular, we make the following contributions:

- We share FEDSZ as an open-source tool integrated with the Advanced Privacy-Preserving Federated Learning (APPFL) package [10], which effectively compresses any PyTorch compatible FL model with EBLC and lossless compression. Our study is the first attempt to reduce communication overhead for FL through EBLC.
- We comprehensively characterize both EBLC and lossless compression algorithms to discover which combination most effectively reduces the size of FL updates while maintaining a low runtime and minimal impact on model capabilities. We find that `SZ2` and `blosc-lz` are the best compressors for low-runtime, data-reduction, and accuracy-preservation from our test suite.
- We evaluate FEDSZ's communication savings by performing rounds of FL on a cluster and simulating different network bandwidths. Experimental results confirm that we achieve (i) 5.55–$12.61\times$ space savings in client updates while keeping uncompressed inference accuracy, (ii) $13.26\times$ communication time savings for a 10Mbps bandwidth,
- We find from the distribution of error introduced by lossy compression near matches to Laplacian distributions, which demonstrates a great potential for lossy compression to introduce differential privacy, an essential security technique for FL.

The organization of our paper is as follows. In Section II, we describe lossy compression's impact on communication overhead to motivate the study. In Section III, we discuss similar work in the field of FL communication cost reduction. Section IV describes our approach through mathematical formulation. Section V describes our design strategies and methods for optimizing FL communications. We present our results in Section VI. We finally conclude the paper with a discussion and future directions in Section VIII.

II. BACKGROUND AND MOTIVATION

In this section we set the stage for how lossy compression can improve I/O efficiency in FL workflows and motivate our proposed FEDSZ approach.

A. Error-bounded Lossy Compression

Lossy compression techniques explicitly designed for floating-point data like `SZ2` [11], `SZ3` [7], [8], `SZx` [12], and `ZFP` [9] enable significant reductions in storage and transmission costs while preserving data accuracy needed for analysis. Below, we summarize four state-of-the-art EBLCs with fundamentally different designs and implementations. The four compressors are also representative of three classic lossy compression models: prediction-based, bit-wise encoding, and transform-based, respectively.

- **`SZ2`** [11] operates on a prediction-based model. It processes datasets in small multi-dimensional blocks, applying hybrid prediction using Lorenzo and linear regression. Post-prediction, `SZ2` quantizes prediction errors and compresses the resultant integers using Huffman encoding and Zstd, ensuring efficient data reduction.
- **`SZ3`** [7], [8] enhances the prediction model of `SZ2` with multi-dimensional dynamic spline interpolation, followed by quantization, Huffman encoding, and Zstd. This approach, not requiring storage of linear regression coefficients, offers improved compression ratios, especially beneficial for higher error bounds.
- **`SZx`** [12], designed for speed, adopts a bit-wise-operation-based encoding model. It segments data into consecutive blocks, determining if each can be represented as a constant block within a given error bound. Non-constant blocks undergo bit-wise truncation. This method's simplicity ensures rapid compression and decompression.
- **`ZFP`** [9], diverging in approach, employs a transform-based model. It applies a custom orthogonal transform to data blocks, encoding the transformed coefficients with specialized bitplane encoders. `ZFP`'s method offers high compression ratios and speeds, benefiting from its optimized transform and encoding strategies.

B. Compression on Networked Systems

The efficiency of data compression in FL, particularly in the context of FEDSZ, hinges on balancing the computational cost of compression and decompression against the time saved in data transmission. To formalize this, we define several key variables: t_C and t_D represent the runtimes for compression and decompression, respectively; S and S' are the original and compressed data sizes; and B_N signifies the network bandwidth. With these variables, we define the inequality in Eqn. 1, which describes the situation where there is a runtime benefit from lossy compression: the total time spent on reduction (t_C), decompression (t_D), and transmitting the compressed data (S'/B_N) should be less than the time to send the original, uncompressed data (S/B_N).

$$0 < t_C + t_D + \frac{S'}{B_N} < \frac{S}{B_N}, \tag{1}$$

This inequality serves as a decision-making criterion, dictating when the benefits of reduced data transmission time outweigh the computational costs of compression. Its relevance

Authorized licensed use limited to: CLEMSON UNIVERSITY. Downloaded on September 19,2024 at 02:45:08 UTC from IEEE Xplore. Restrictions apply.

to FEDSZ lies in its ability to guide the algorithm toward 'worthwhile' reduction, optimizing the trade-offs in scenarios with limited bandwidth and computational resources.

These compression and communication challenges motivate solutions to efficiently manage and transfer data in FL without sacrificing model performance. Our proposed compression approach called FEDSZ, described in later sections, directly addresses these needs to improve communication efficiency for the resource-constrained federated environment.

III. RELATED WORKS

In this section, we discuss three aspects of related work to identify the gap FEDSZ aims to address while foreshadowing the advantages of our methodology.

A. Impact of Lossy Compression on Data Transfer

EBLC has been widely explored to improve data transfer performance significantly in many scenarios. For example, Liang et al. [13] found that various supercomputers may possess different I/O bandwidths in practice, and that selecting appropriate lossy compressors can be critical for maximizing the overall I/O performance. Specifically, Liang et al. characterized the I/O bandwidth on multiple supercomputers and developed an adaptive lossy compression framework combining `SZ` and `ZFP` to maximize the I/O performance. Liu et al. [14] developed *Ocelet*, which can leverage parallel EBLC to accelerate the data transfer performance between Globus endpoints on a wide area network (WAN). Over 90% of the transfer time can be reduced by Ocelet. Compared with the above existing use cases, using EBLC in FL faces more challenges such as lower network bandwidth between clients and servers and the parameter data being much less compressible than scientific datasets. Therefore, there is a gap in the known literature concerning how effective EBLC is in both edge case scenarios and FL scenarios.

B. Compression of Machine Learning Models

Several techniques have been proposed for compressing deep neural networks (DNNs), including lossless and lossy methods. Lossless compression approaches like lossless expressiveness optimization [15] use linear programming and rectified linear units to encode network weights; however, this method can have low compression ratios and high compression time overheads. Lossy DNN compression techniques, such as DeepSZ [16] and Delta-DNN [17], first apply pruning to sparsify networks and then quantization to shrink weight parameters. These lossy methods can often degrade model accuracy and/or require costly retraining to recover performance. In FL this is not tenable, as communication rounds can be infrequent and expensive for battery-constrained and distributed devices. Universal randomized lattice quantization [18] has been proposed to compress DNNs by quantizing weight vectors, offering high compression ratios regardless of the distribution of parameters, with the limitation of low-granularity quantization. Therefore, it is necessary to explore EBLC's effects on model accuracy in FL settings to reduce the communication time overhead with high compression ratios while maintaining uncompressed accuracy.

C. FL Communication Cost Reduction

In FL, various compression methods have been explored to enhance communication efficiency. Gradient sparsification, as initially proposed by Strom (2015), involves transmitting only significant gradients, with recent advancements favoring dynamic thresholds for broader applicability [19]–[21]. The Top-K method, a popular technique, selects the largest gradients for transmission, ensuring efficient parameter updates [22]–[25]. Gradient quantization, exemplified by one-bit SGD and signSGD, reduces data transmission size but often results in biased estimates [26], [27]. Recent methods like TerGrad and QSGD employ stochastic unbiased estimations to maintain accuracy [28], [29]. Low-rank approaches, leveraging the inherent properties of over-parameterized DNN models, focus on efficient gradient matrix decomposition [30]–[34]. Comparison to the aforementioned existing methods is difficult or not possible as they are not open-source. Therefore, any meaningful comparison would require reimplementation, which is outside of the scope of this exploration.

While these methods offer potential benefits in FL, EBLC is able to reconstruct a dense network of weights with the original floating-point precision. FEDSZ is a "last-step" in the communication pipeline, meaning that it is capable of further compressing sparsely trained or quantized model updates. Therefore, comparison to existing methods is not necessary, as any method can ostensibly can be used in concert with FEDSZ. EBLC, through its reconstruction approach to compression and decompression, can help address the adjustments in training and sparsity that can introduce bias and inaccuracy in compressed models.

IV. PROBLEM FORMULATION

In this section, we present the optimization framework that guides the design of FEDSZ. The framework addresses two multi-objective optimization problems, aiming to improve both computational efficiency and model performance in an FL environment.

A. Problem 1: Lossy Compressor Selection

Our first problem focuses on selecting an optimal EBLC from a set, $\mathcal{X}_{\text{lossy}}$. The objective is twofold: minimize the time overhead and maximize the compression ratio. These objectives are crucial as they directly affect the efficiency and speed of the FEDSZ process. Each compressor $x \in \mathcal{X}_{\text{lossy}}$ is associated with error bound ϵ, a critical parameter that impacts both the compression ratio $R(x, \epsilon)$ and the runtime $T(x, \epsilon)$.

The constraints are specifically chosen to ensure practicality and efficiency. The feasible region for R is $[1, S]$, with S being the original number of elements and 1 being no compression. This range ensures that the compression remains advantageous for a user. For the time T, the feasible region is $\left(0, \frac{S}{B_N}\right)$, where B_N is the network throughput, thus ensuring that the time overhead doesn't exceed network capabilities.

Authorized licensed use limited to: CLEMSON UNIVERSITY. Downloaded on September 19,2024 at 02:45:08 UTC from IEEE Xplore. Restrictions apply.

$$x^* = \arg\left\{ \max_{x\in\mathcal{X}_{\text{lossy}},\epsilon>0} R(x,\epsilon), \min_{x\in\mathcal{X}_{\text{lossy}},\epsilon>0} T(x,\epsilon) \right\}$$
$$\text{subject to} \quad 0 < T(x,\epsilon) < \frac{S}{B_N}, \qquad (2)$$
$$1 \le R(x,\epsilon) \le S.$$

B. Problem 2: Compression in Federated Learning

Building upon Problem 1, our second research problem focuses on effectively integrating our optimal compressor x^* into the FEDSZ algorithm. The choice of x^* directly influences the computational overhead, communication reduction, and accuracy impacts in our distributed FL setting, making it a cornerstone for this problem.

Here, we suppose there are n clients, and each client-i has an associated communication costs P_i and network bandwidth $B_{N,i}$. The error bound variable ϵ is our critical factor, affecting both the communication cost $P_i(\epsilon)$ and the inference accuracy $I(\epsilon)$ at the server during aggregation and validation. Parameter $I' \in [0,1]$ serves as the baseline accuracy when no compression is applied, and the goal is to find the optimal ϵ^* that minimizes the discrepancy between I' and $I(\epsilon)$.

The constraints are set to ensure that the solutions are both feasible and effective. The error bound ϵ is constrained to be greater than zero to avoid trivial solutions. The communication cost $P_i(\epsilon)$ is bounded by the network bandwidth, and the accuracy $I(\epsilon)$ is naturally bounded between 0 and 1 to represent valid probability measures.

$$\epsilon^* = \arg\left\{ \min_{\epsilon>0} \sum_{i=1}^{n} P_i(\epsilon), \ \min_{\epsilon>0} |I' - I(\epsilon)| \right\}$$
$$\text{subject to} \quad 0 \le I(\epsilon) \le 1, \qquad (3)$$
$$0 \le P_i(\epsilon) \le \frac{S_i}{B_{N,i}}.$$

Formulations (2) and (3) provide the mathematical foundation for the FEDSZ algorithm. These equations tackle the computational and communication complexities in FL, thereby laying the groundwork for the subsequent development and analysis of the FEDSZ algorithm.

V. FEDSZ: A FEDERATED LOSSY COMPRESSION SCHEME

Our main contribution in this paper is FEDSZ: a generally applicable compression scheme for FL client-server communications. Our technique is summarized as follows: (i) partitioning a client update (represented as a PyTorch model state dictionary) into lossy and lossless components, (ii) lossy and lossless compression of the partitions, and (iii) communication of the bitstream for decompression at the receiving host. This method is illustrated in Figure 1, which involves the compression stage (as shown in Figure 1(a)) and decompression stage (as shown in Figure 1(b)). The rest of this section details the different considerations of our design and our strategies to solve our outlined research problems from Section IV.

A. Characterizing Model State Data

First, we characterize the shape and distribution of FL model weights in this subsection, something that is critical to understanding the challenges of using EBLC to compress model data. We compare the value variation of FL model parameters versus the classic scientific simulation data in Figure 2, where the relative data index means the 1D data index in the specific snippet or slice. The figure illustrates that the FL model parameters are very spiky, while the classic simulation datasets exhibit much higher smoothness. This is because the scientific simulation data are often used to convey specific physical or chemical phenomena, as demonstrated in Figure 2 (c) and (d). As such, a serious question arises: Can the EBLCs still work effectively on FL model parameters? Moreover, which compressor is the best choice? We answer them in our study.

B. Overview of FEDSZ Algorithm

We design FEDSZ based on observations from careful analysis and experiments. Specifically, we learn that our compression scheme, summarized in Figure 1, should include three strategies: (i) partitioning the client update into lossy and lossless components, (ii) using `SZ2` to lossy compress and `blosc-lz` to lossless compress, and (iii) dumping a bitstream to communicate the client update to the server. We describe the design details in the Algorithm 1.

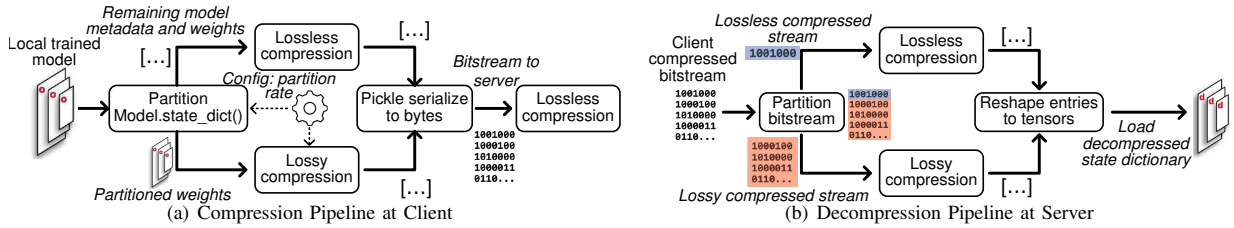
Algorithm 1 FEDSZ Compression Scheme

```
Require: model: torch.Module, threshold: int
Ensure: compressed_model: bytes
1:  Initialize: compressed_params = {}
2:  for each (name, param) in model.state_dict() do
3:      flat_param ← param.flatten()
4:      if "weight" in name and flat_param.size > threshold then
5:          compressed_param ← lossy_compress(flat_param)
6:      else
7:          compressed_param ← lossless_compress(flat_param)
8:      end if
9:      compressed_params[name] ← compressed_param
10: end for
11: return compressed_params.to_bytes()
```

C. Partitioning the State Dictionary

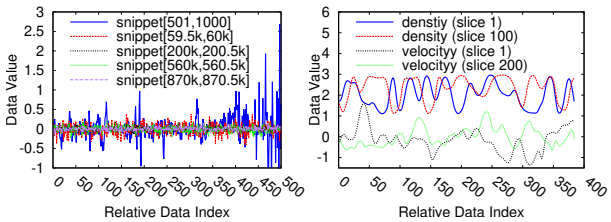
Our FEDSZ algorithm employs a partitioning strategy dividing the state dictionary into the components that we can compress without sacrificing accuracy and the parts that require lossless compression to maintain the integrity of the model state (as shown in Line 4 of Algorithm 1). The complexities in efficiently compressing DNNs necessitate a method for partitioning the model's state dictionary into lossy and lossless compressible segments (see line 4-8). In a DNN, maintaining a model for training and evaluation involves various layers and values, including parameter tensors, running means, and bottlenecks. The `state_dict()` captures the complete state of the model, both trainable and non-trainable. However, lossy compression of both parameters/weights and metadata risks significant loss of important values and extreme degradation of model accuracy, which has we have experimentally verified. Our observation is also consistent with DeepSZ [16], which

Authorized licensed use limited to: CLEMSON UNIVERSITY. Downloaded on September 19,2024 at 02:45:08 UTC from IEEE Xplore. Restrictions apply.

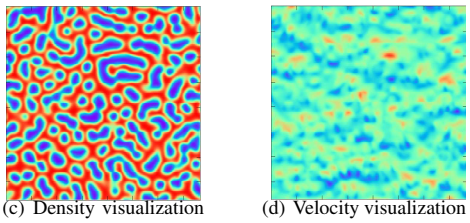


(a) Compression Pipeline at Client (b) Decompression Pipeline at Server

Fig. 1: FEDSZ Design for both Compressing and Decompressing Local Model Updates



(a) Snippets of FL Model Parameters (b) Snippets of Miranda Data



(c) Density visualization (d) Velocity visualization

Fig. 2: Comparing FL Model Parameters vs. Scientific Simulation Data (MIRANDA [35], [36])

aims to save storage space for ML models using lossy compression. As such, we adopt an selective compression strategy to partition models into the dense, compressible weights and non-compressible metadata.

D. Exploring the Most Effective Lossy Compression Method

In our exploration of determining which lossy compressor we should use, we compare `SZ2` (v1.12.5), `SZ3` (v3.1.7), `SZx` (v1.0.0), and `ZFP` (v1.0.0), all EBLCs with different characteristics as described in Section II-A. To decide which EBLC to use, we carefully investigate (i) the distribution of the data to compress, (ii) the impact on inference accuracy, and (iii) the performance of the compressors. All runtime and throughput data are computed on a Raspberry Pi 5 with 8GB of RAM. All training and inference accuracy data are computed on Argonne's Swing cluster, where each node has 8×NVIDIA A100 40GB GPUs and 2×AMD EPYC 7742 64-Core Processors with 128 cores.

1) Relative Error Bounding: Selecting an appropriate error bounding mode is a nuanced task that directly correlates with the statistical characteristics of the data to be compressed [37]. Our study focuses on the weight distributions of three distinct models: AlexNet, MobileNetV2, and ResNet50, as visualized in Figure 3. While each model's weight distribution is between -1 and 1, they exhibit different dynamic ranges and clustering behaviors around zero.

Given these variations in weight distributions, a fixed error bound could either be overly conservative for some data

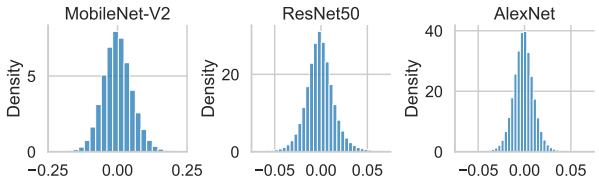


Fig. 3: Distribution of Pretrained Weights for Various Models

segments or too liberal for others, thereby not exploiting the compression potential to its fullest. This is where the utility of relative error bounds becomes apparent. A *relative error bound* adapts to the local properties of the data by being a multiple of the dynamic range, ensuring that the error introduced during compression is proportional to the data's actual value. Consequently, we opt for relative error bounding modes for `SZ2`, `SZ3`, and `SZx`. For `ZFP` there is not a relative error mode, so we select the closest analogous option, which is fixed precision mode, where the number of uncompressed bits is fixed. This choice allows us to harness the benefits of adaptive error control, accommodating the dynamic ranges observed across different layers in a model.

2) Inference Accuracy Convergence Comparison: The measure of the efficacy of a lossy compression algorithm in an FL setting such as FEDSZ is not just compression performance but also its impact on the model's inference accuracy and convergence behavior. Achieving a high compression ratio is of little benefit if it comes at the cost of a model that fails to converge to a satisfactory level of accuracy. Therefore, in addition to comparing raw compression metrics, we also closely examine how each compressor influences the model's accuracy throughout communciation rounds. We run an FL simulation on a cluster of training AlexNet for the CIFAR-10 task for ten rounds with one epoch per round while compressing the updates with each candidate compressor. We note that results for other models and datasets are similar as it is essentially compression of "spiky" 1D floating-point data, therefore other results are not included. The observed trends in inference accuracy for these compressors are depicted in Figure 4.

As illustrated, the accuracy convergence for most compressors are closely aligned, indicating a minimal difference in compressor impact. The outlier compressor for accuracy retention is `SZx`, which fails to maintain any accuracy, likely due to its block mean storage. Our findings affirm that a nu-

Authorized licensed use limited to: CLEMSON UNIVERSITY. Downloaded on September 19,2024 at 02:45:08 UTC from IEEE Xplore. Restrictions apply.

TABLE I: EBLC Comparison Across Different Models for CIFAR-10–Throughput refers to an EBLC's data processing rate

Model	Compressor	Runtime (s)			Throughput (MB/s)			Compression Ratio			Top-1 Accuracy (%)		
		Relative Error Bound											
		10^{-2}	10^{-3}	10^{-4}	10^{-2}	10^{-3}	10^{-4}	10^{-2}	10^{-3}	10^{-4}	10^{-2}	10^{-3}	10^{-4}
AlexNet	SZ2	3.22	4.93	6.64	70.75	46.26	34.34	**11.26**	**5.228**	**3.375**	**57.90**	**57.39**	57.06
	SZ3	7.22	8.80	10.69	31.58	25.94	21.34	9.827	5.169	3.345	57.28	57.18	**58.50**
	SZx	**0.444**	**0.439**	**9.445**	**3514.92**	**3554.84**	**3507.02**	4.804	4.801	4.802	10.00	10.00	10.00
	ZFP	1.89	2.11	2.36	120.66	108.17	96.51	4.166	2.942	2.210	57.30	56.80	57.19
MobileNet-V2	SZ2	0.365	0.482	0.692	25.61	19.38	13.51	**5.409**	3.235	1.923	**55.19**	55.19	**55.19**
	SZ3	0.574	0.724	1.082	16.30	12.92	8.64	5.250	3.125	1.779	54.94	56.14	53.86
	SZx	**0.034**	**0.031**	**0.029**	**104.089**	**114.162**	**122.036**	4.799	**4.765**	**4.783**	10.00	10.00	10.00
	ZFP	0.127	0.138	0.151	73.73	67.57	61.70	3.027	2.333	1.897	54.600	**57.26**	54.90
ResNet50	SZ2	1.235	1.930	2.870	76.80	49.14	33.05	**7.025**	4.041	2.737	58.66	59.15	58.66
	SZ3	2.723	3.459	4.586	34.82	27.42	20.68	6.768	3.942	2.662	**64.14**	61.35	63.51
	SZx	**0.186**	**0.186**	**0.185**	**3516.02**	**3516.02**	**3535.27**	4.806	**4.806**	**4.806**	10.00	10.00	10.00
	ZFP	0.747	0.843	0.962	126.95	112.53	98.58	3.449	2.562	2.035	62.07	**64.70**	**64.06**

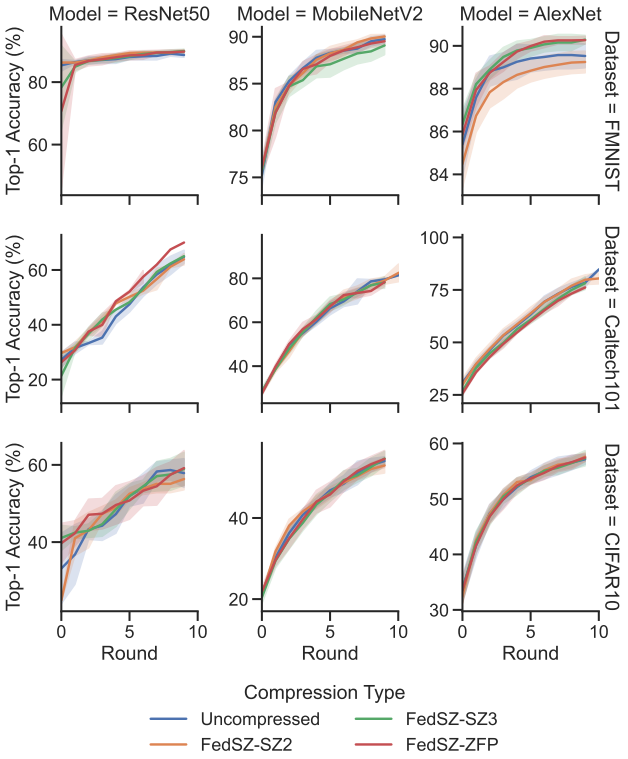


Fig. 4: Accuracy Convergence Comparison for EBLCs

anced balance between compression performance and accuracy impact is critical for the practical applicability of the chosen lossy compressor.

3) Performance Characteristics: We summarize our comparison of EBLCs for FEDSZ in Table I when using a Raspberry Pi 5 for model compression. From these results, we find that SZ2 is the optimal compressor. This table summarizes key metrics across multiple error bounds. The accuracy metrics has been evaluated on a cluster while training the AlexNet, MobileNet-V2, and ResNet50 models on the CIFAR-10 dataset over ten communication rounds, and the runtime and throughput data is from compressing the trained models on a Raspberry Pi 5. There was no impact on training time besides the incurred overhead of compression, as will be discussed in Section VII-B.

Our primary focus is discerning which EBLCs achieve good compression ratios with a low runtime and then maintain accuracy similar to not using compression. In this regard, SZ2 stands out. It outperforms SZ3 and ZFP in compression ratios by 14.6% and 170.3%, respectively for AlexNet. Notably, this comes at the relatively modest increase in runtime when compared to ZFP, only 3.1% more per epoch. SZ2 continues to demonstrate a strong balance between high compression ratios and maintaining high accuracy across different models. For MobileNet-V2, at an error bound of 1E-2, SZ2 achieves a compression ratio of 5.409 and a Top-1 Accuracy of 55.19%, outperforming SZ3 and ZFP in both metrics. Similarly, for ResNet50, SZ2 achieves a compression ratio of 7.025 and maintains a Top-1 Accuracy of 58.66%, surpassing its counterparts. For reducing I/O overhead, a higher compression ratio can be more valuable than marginal gains in runtime [38], and SZ2's modestly greater runtime is a justifiable tradeoff for its superior compression ratio.

The justifications for these results are as follows. Since FL model parameters are flattened to 1D data arrays with an unpredictable distribution, (as demonstrated in Section 2) after which we have to perform 1D lossy compression. ZFP is optimized particularly for multi-dimensional datasets but may perform poorly on 1D spiky datasets, as disclosed by prior literature [39]–[41]. Thus, it achieves relatively low compression ratios on FL model parameters. SZx [12] uses relatively simple compression operations (such as marking constant blocks and using bit-wise operations) as it was designed primarily to have low compression/decompression runtimes, which, may cause poor compression ratios and reconstructed data quality. SZ2 and SZ3 should exhibit similar compression ratios because they both default to using a Lorenzo predictor and quantization when data exhibit significant variations, according to their hybrid compression design [7], [11]. In comparison, SZ3 features a more sophisticated predictor selection policy, leading to lower compression throughput. All in all, SZ2 exhibits the best tradeoff in terms of runtime, compression ratio, and data reconstruction quality, which we verify through our result in Table V.

Authorized licensed use limited to: CLEMSON UNIVERSITY. Downloaded on September 19,2024 at 02:45:08 UTC from IEEE Xplore. Restrictions apply.

Our method for selecting SZ2 as our preferred compressor aligns coherently with the design criteria and feasibility region outlined in Eqn. (2). Here, we are guided by the dual objectives of achieving a low runtime and a high compression ratio, all within a predefined feasibility region. The empirical data solidifies SZ2 as the most suitable lossy compressor for FEDSZ, effectively balancing performance metrics with the practical constraints of an FL environment.

E. Exploring Best Fit Lossless Compression for Metadata and Non-weight Parameters

In the FEDSZ algorithm, we use lossless compression to reduce the size of client updates' metadata and non-weight parameters ($\approx 1\%$ of an update's storage size). Similar to the lossy case, we evaluate several state-of-the-art lossless compressors. The considered compressors include blosc-lz (v1.21.3) [42], zlib (v1.2.13) [43], zstd (v1.5.5) [44], gzip (Python v3.11.4) [45], and xz (v0.5.0) [46]. To compare these options, we compress the metadata and non-weight parameters for AlexNet to evaluate the compressors' performances, including the results in Table II. Since lossless compressors do not introduce noise into the client updates, we do not need to check if there are impacts on inference accuracy.

TABLE II: Lossless Compressor Comparison for Compressing AlexNet Metadata on Raspberry Pi 5

Compressor	Runtime (s)	Throughput (MB/s)	Compression Ratio
blosc-lz	**0.271**	**674.5**	1.248
gzip	7.728	28.16	1.160
xz	74.52	4.00	**1.250**
zlib	7.772	28.37	1.164
zstd	0.529	348.6	1.169

From Table II, we notice that blosc-lz outperforms the other lossless compressor by achieving more than $2\times$ lower runtime than zstd and a comparable compression ratio to xz, a very slow lossless compressor. Taking these two points, it is clear to see that blosc-lz is a fair choice for our metadata. Just like the data distribution mentioned in Section V-A, it is important to note that the target data are formatted as floating-point, small-size 1D arrays of non-uniform data, which will lead to low compressibility.

VI. METHODOLOGY FOR EVALUATING FEDSZ

With the design of FEDSZ documented, we now detail our evaluation methodology.

A. Federated Learning Platform: APPFL with FedAvg

We implement FEDSZ in an open-source Python package, APPFL (v0.4.0) [10], a library designed for the development and evaluation of privacy-preserving FL (PPFL) algorithms. APPFL offers modular APIs for implementing essential components such as learning algorithms, privacy mechanisms, communication protocols, models, and data handling. The package utilizes gRPC for cross-platform communication and employs Message Passing Interface (MPI) through mpi4py [47] for parallelism to enable scalable simulations. It is compatible with PyTorch, allowing for the integration of custom neural network models. APPFL has been successfully used to train models across various decentralized datasets, including biomedical images [48] and electric grid [49], [50].

In FL, the package provides the capability to train a global model by aggregating updates from client models operating on decentralized data. Various algorithms can be employed, each with different privacy, efficiency, and robustness trade-offs. For the scope of this study, we focus on Federated Averaging (FedAvg) [51], a well-established algorithm that performs local Stochastic Gradient Descent (SGD) on client devices and averages these local models to update a global one. The choice of FedAvg is deliberate; its simplicity, scalability, and robust performance make it compatible with compression techniques, thus serving our study's objectives effectively.

B. Models and Datasets for Training

The primary aim of FEDSZ is to optimize client-server communication efficiency without sacrificing model accuracy or imposing a significant compression runtime overhead. To rigorously evaluate our framework, we chose DNNs that vary significantly in terms of parameter count, model size, percentage of data that we lossy compress, and computational complexity (FLOPs). The characteristics of these models are summarized in Table III.

TABLE III: DNNs for FEDSZ Profiling: Mean Statistics

Model	Parameters	Size	% Lossy Data	FLOPs
MobileNet-V2 [52]	$3.5e+06$	14MB	96.94%	0.35G
ResNet50 [53]	$4.5e+07$	180MB	99.47%	8G
AlexNet [54]	$6.0e+07$	230MB	99.98%	0.75G

Selecting a diverse set of models ensures that our results are not specific to any particular architecture, making our insights broadly applicable. For instance, MobileNet-V2, with its relatively fewer parameters and FLOPs, represents edge cases where the device capabilities might be limited. In contrast, ResNet50 offers a more complex architecture suitable for resource-rich environments, and AlexNet serves as a middle-ground model.

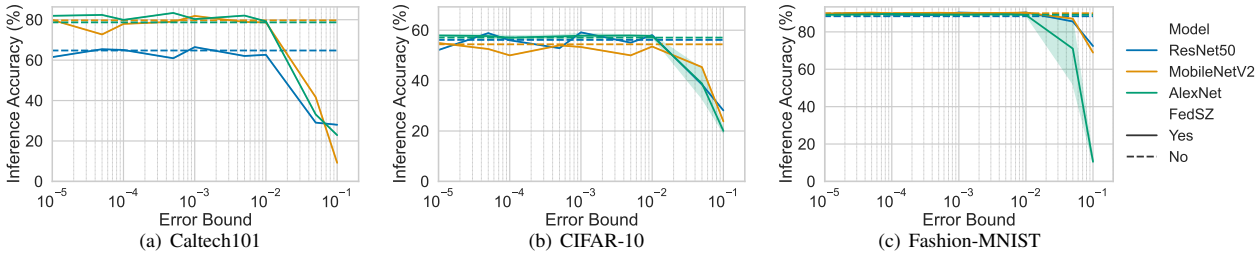
Similarly, we chose three well-established image classification datasets to maintain comparable tasks across different models, as detailed in Table IV.

TABLE IV: Dataset Characteristics for FEDSZ Benchmarking

Dataset	# of Samples	Input Dimension	Classes
CIFAR-10 [55]	$60,000$	32×32	10
Fashion-MNIST [56]	$70,000$	28×28	10
Caltech101 [57]	$9,000$	224×224	101

The choice of CIFAR-10, Fashion-MNIST, and Caltech101 serves multiple purposes. CIFAR-10 and Fashion-MNIST are more straightforward datasets, often used for benchmarking. They allow us to assess how well FEDSZ performs under less demanding conditions. Caltech101, with its higher image pixel count and greater number of classes, is a more challenging

Authorized licensed use limited to: CLEMSON UNIVERSITY. Downloaded on September 19,2024 at 02:45:08 UTC from IEEE Xplore. Restrictions apply.



(a) Caltech101 (b) CIFAR-10 (c) Fashion-MNIST

Fig. 5: Inference Accuracy Across Diverse Models and Datasets while Varying FEDSZ Relative Error Bound

image classification problem, testing that the error introduced by our compression doesn't just work with "toy" tasks. These datasets provide a comprehensive test bed for evaluating FEDSZ across various scenarios. We perform our testing using FedAvg for four clients and fifty communication rounds with one epoch per client per round.

C. Simulating Network Bandwidth

To evaluate FEDSZ's communication reduction in low bandwidth network conditions, we emulate different bandwidths by introducing delays into MPI when sending data between processes. We make a note that part of FEDSZ's evaluation is on a cluster where we use MPI in APPFL for simulating multiple FL clients, therefore this simulation is from our cluster deployment and not the Raspberry Pi 5.

The goal of this method is to demonstrate how FEDSZ would perform with low bandwidth, which is common in real-world FL deployments. We are able to simulate this effect by first measuring the process-to-process MPI bandwidth. Then, we use `sleep` to wait for the length of time it would take to transmit our compressed client update proportional to our desired bandwidth. By incorporating these controlled sleep delays into the communication protocol, we simulate the low-bandwidth environments typical of edge devices (e.g., 10Mbps [58]) compared to the data center setting which can approach 10Gbps. This strategy allows us to demonstrate the communication runtime reduction offered by FEDSZ's compression techniques.

VII. PERFORMANCE EVALUATION

Having outlined the guiding design for FEDSZ and our evaluation methodology, we present results demonstrating the impact of FEDSZ on the inference accuracy, runtime, and communication time of FL and the scalability of FL with FEDSZ through strong and weak scaling. In our evaluation, we prototype rounds of learning on the Argonne Swing cluster to accelerate learning tasks and measure compression runtime results from a Raspberry Pi 5. Therefore, for our experiments showing the learning and accuracy capabilities of FEDSZ during training, we use the system described in Section V-D and for runtime and compression benchmarking we use data collected from a Raspberry Pi 5.

A. Compression Impacts on Inference Accuracy

An important consideration in our study is the trade-off between compression ratio and model accuracy when utilizing FEDSZ with `SZ2` and `blosc-lz` compressors. Therefore, we test the impact of varying `SZ2`'s relative (REL) error bounds from 10^{-5} to 10^{-1}. Our findings, illustrated in Figure 5, reveal a clear threshold at 10^{-2} beyond which model accuracy sharply declines due to the magnitude of the error introduced by the compressor. This decline indicates a boundary where a loss in model utility offsets the benefits of higher compression (and thus reduced communication costs). However, for error bounds less than or equal to 10^{-2}, we observe that the inference accuracy remains within 0.5% of the uncompressed model's performance. At some error bounds, there is a deviation from this 0.5%. However, this is due to the natural variability of training and validation. This is a crucial observation, underscoring that we can compress FL client updates without affecting a model's efficacy.

Moreover, the compression ratios presented in Table V further corroborate our recommendation of 10^{-2} as an optimal REL. For instance, at a REL of 10^{-2}, AlexNet achieved a compression ratio of $12.61\times$ on CIFAR-10, a significant gain without damaging model accuracy. We observe similar trends for MobileNetV2 and ResNet50 across all datasets. The consistency in these trends across various model architectures and datasets suggests that a REL of 10^{-2} could be a generalized optimal setting in a wide range of FL applications. Similarly, as shown in Figure 4, we see that FEDSZ did not significantly impact the rate of convergence for models. The shading in this plot represents the standard deviation of accuracy over all the explored relative error bounds $< 10^{-2}$, revealing that there is no impact on convergence for error bounds less than this value.

Taking into account Eqn. 3, we further the argument that choosing of error bound is a critical factor in achieving an effective balance between compression efficiency and model accuracy. Therefore, based on our results, we recommend a relative error bound of 10^{-2} as it offers communication runtime reduction without compromising model performance, making it well-suited for FL optimizing client-server communications.

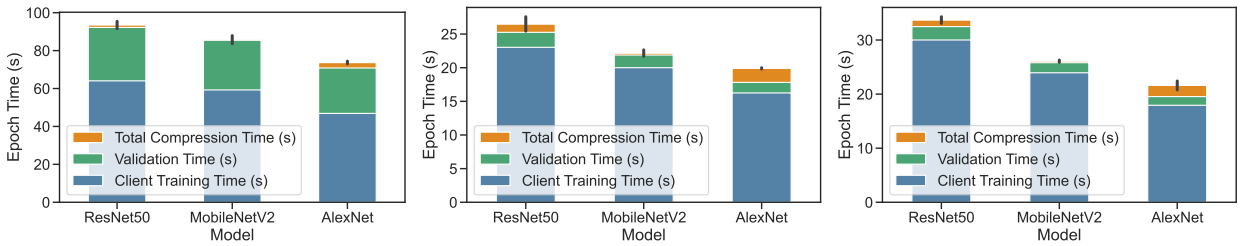
B. Time Overhead of FEDSZ and Network Gains

1) Cluster Setting: A primary concern of FEDSZ is whether the runtime overhead introduced by compressing and decompressing model updates outweighs the gains in reduced communication runtime (see Eqn 1); however, we find that the advantages of FEDSZ in communication runtime reduction outweighs the introduced compression overhead. For a cluster

Authorized licensed use limited to: CLEMSON UNIVERSITY. Downloaded on September 19,2024 at 02:45:08 UTC from IEEE Xplore. Restrictions apply.

TABLE V: Compression Ratios for FEDSZ for Various Models and Datasets

Dataset	CIFAR-10				Caltech101				Fashion-MNIST			
REL Error Bound	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-1}	10^{-2}	10^{-3}	10^{-4}
AlexNet	54.54	12.61	5.54	3.52	51.19	9.62	4.94	3.27	186.42	12.01	5.69	3.57
MobileNetV2	11.07	5.39	3.23	1.94	10.62	5.26	3.18	1.93	11.70	5.55	3.27	1.98
ResNet50	20.21	7.02	4.04	2.73	17.81	6.68	3.90	2.66	22.76	7.11	4.12	2.74



(a) Caltech101 (b) CIFAR-10 (c) Fashion-MNIST

Fig. 6: Client Runtime per Epoch Breakdown including FEDSZ Compression

setting, Figure 6 shows the mean runtime per epoch across various models and datasets when using FEDSZ with an error bound of 10^{-2}. The compression process takes additional runtime, at its most extreme case with AlexNet on CIFAR-10, the overhead was 3.66 seconds or 17% of the total runtime. In the rest of the cases, the wall-clock overhead was $<12.5\%$, and an average of 4.7% of the client's total epoch time. These times, however, are relatively minor when evaluated in the context of the overall communication time savings that compression offers, particularly at larger error bounds.

Figure 7 shows the simulated communication time of a client update to the server, including compression and decompression, on a network of 10Mbps; when we compress at any error bound, we decrease the communication time by an order of magnitude from uncompressed data transmission. For example, at an error bound of 10^{-2} on a simulated 10Mbpsnetwork, AlexNet experienced a 109.87s or $13.26\times$ reduction in communication time. Similarly, reductions for MobileNetV2 and ResNet50 were by 12.23% and 9.74%, respectively. This means that the server receives updates faster and can begin aggregating the results to begin the next communication round, and the client is using less runtime on I/O as previously.

2) Edge Setting: The merit of employing FEDSZ is clear when examining the communication time savings in a limited network bandwidth setting. Since FL is typically decentralized learning on geographically distributed edge devices, we include benchmarks of the compression overhead when using FEDSZ on a Raspberry Pi 5 in Table I. Some quantities, such as accuracy and compression ratio, are hardware independent, so a key metric then is how long compression takes on a certain system. The importance of this point is made clear in Eqn. 1, where we see that compression for communications is advantageous with a low compression runtime and high compression ratio. We should note that this is irrespective of and has no effect on training time, as FEDSZ is post-training.

An interesting consequence of Equation 1 is it reveals whether you should compress for a given network bandwidth. Using the runtime overhead and compression ratios from Tables I and V, respectively, we can calculate the communication runtime for a spectrum of bandwidths to generate Figure 8.

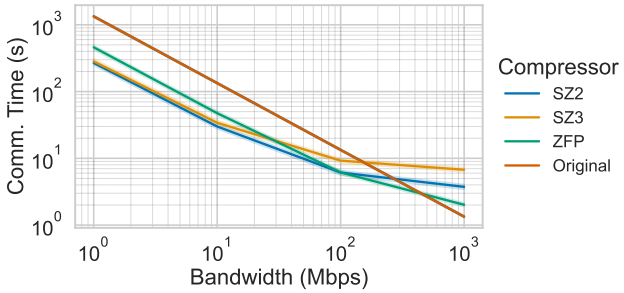


Fig. 8: Communication Time for Transmitting AlexNet over Variable Network from Raspberry Pi 5

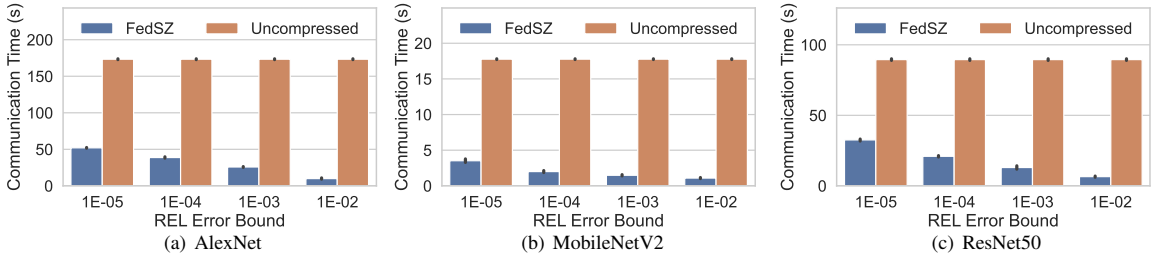
Here we notice that until approximately 350Mbps compression is optimal, and until 100Mbps SZ2 is most optimal, before the runtime overhead of compression takes outweighs the gains in compression ratio. Therefore on a < 350Mbps wide area network with edge device compression latency, it is more runtime efficient to compress before communication. The gains in communication reduction before this threshold are significant before this point, as a client can save an order of magnitude of runtime when transmitting a model update to the server.

C. Scalability of FEDSZ

To test whether FEDSZ scales effectively with the number of clients in a system we evaluate scalability up to 128 CPU cores on Argonne's Swing cluster, increasing the CPU core count in powers of 2 while simulating a network bandwidth of 10Mbps using the method described in Section VI-C.

For our weak scaling test, we assign one client per core while increasing the total number of CPU cores. FEDSZ demonstrates effective weak scaling, as evidenced by the near-

Authorized licensed use limited to: CLEMSON UNIVERSITY. Downloaded on September 19,2024 at 02:45:08 UTC from IEEE Xplore. Restrictions apply.

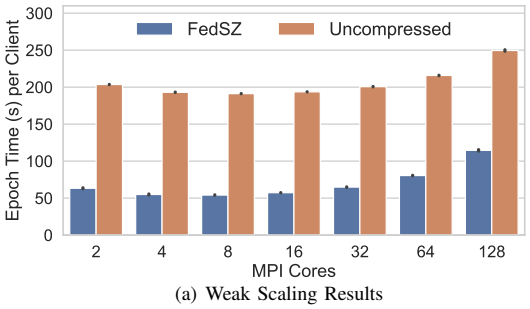


(a) AlexNet (b) MobileNetV2 (c) ResNet50

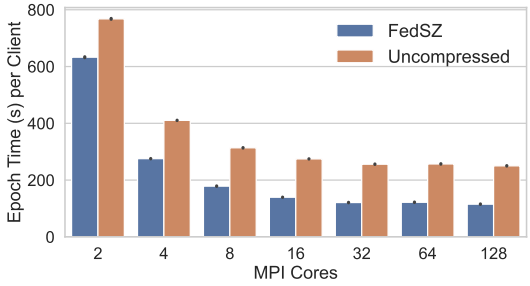
Fig. 7: Total Communication Time for Models over Different REL Error Bounds on 10Mbps Network

linear increase in communication time as the number of cores and clients increased, shown in Figure 9(a). In our evaluation, FEDSZ exhibits a recalculated weak scaling speedup ranging from 0.36 at 128 MPI processes to a peak of 1.64 with 8 MPI processes, indicating moderate adaptability to an increasing client count.

For our strong scaling test, we kept 127 clients constant while increasing the total number of CPU cores. FEDSZ shows robust, strong scaling characteristics as shown in Figure 9(b). In this experiment, the framework achieves a recalculated speedup as high as 7.51 at 128 MPI processes, demonstrating effective utilization of additional computational resources for a fixed number of clients.



(a) Weak Scaling Results



(b) Strong Scaling Results

Fig. 9: Scaling Results for Training MobileNet-V2 on CIFAR-10 with and without FEDSZ

Simulating a constrained network environment of 10Mbps demonstrates the efficacy of FEDSZ's compression algorithms. Under these conditions, the framework is able to maintain the model's accuracy while significantly reducing the communication overhead. The advantage of using compression becomes increasingly evident as the number of cores (and thus, the number of clients) increases. This suggests that FEDSZ's compression techniques can offer substantial benefits in low-bandwidth scenarios, where communication costs are often high.

D. Potential Differential Privacy of Lossy Compression

An intriguing aspect of FEDSZ is that lossy compression adds some noise to the data after decompression, a phenomenon that could potentially introduce to differential privacy (DP). A user can find the distribution of errors by taking the pairwise difference of the original weights and the decompressed weights. Our experiments have shown that error distributions of the communicated model parameters exhibit characteristics similar to a Laplacian distribution, as evidenced by the histograms in Figures 10.

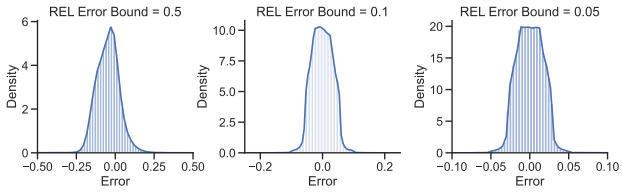


Fig. 10: Distribution of Errors for Different Error Bounds

DP can work by adding noise with a certain random distribution to data before communication (e.g. Laplacian distributed noise) to ensure that an individual's data cannot be easily identified from aggregate statistics [59]. It is worth noting that the resemblance to Laplacian noise does not guarantee DP, as the formal guarantee depends on specific mathematical conditions related to the data sensitivity and the noise scale. However, the observed characteristics in our study make this an avenue worth exploring. Chen et al. [60] corroborate these claims with their findings that types of compression can introduce DP, further meriting exploration into whether EBLCs possess the same properties. Given the critical importance of data privacy in FL, this warrants further investigation and could add another layer of utility to FEDSZ's already promising performance.

VIII. Discussion and Future Work

A. Takeaways and Addressed Questions

- **How to integrate lossy compressors in FL communication?**

Authorized licensed use limited to: CLEMSON UNIVERSITY. Downloaded on September 19,2024 at 02:45:08 UTC from IEEE Xplore. Restrictions apply.

  - We develop FEDSZ, a robust algorithm that seamlessly integrates lossy compression with `SZ2` and `blosc-lz` into the FL client-server communication pipeline. Our implementation supports various model architectures and datasets, offering a modular approach for FL applications.
- **What EBLC would perform best given the task of compressing FL client updates?**
  - Through empirical evidence, we find that `SZ2` consistently outperforms other compressors with respect to compression ratio while maintaining uncompressed model accuracy. It offers a balance between compression efficiency and minimal impact on model performance.
- **EBLCs introduce error and time overhead. Is it worth using this as a data reduction strategy?**
  - Our results indicate that for relative error bounds up to 10^{-2}, FEDSZ maintained model accuracy within 1% of the uncompressed model while achieving significant compression ratios. Thus, the minor time overhead of compression is justifiable given the substantial reduction in communication cost.

B. Future Directions

This study presents a robust compression algorithm for reducing the communication overhead of client-server interactions for FL. Future research directions beyond the scope of this study are to explore (1) how hyperparameter optimization might be tuned to mitigate the accuracy loss introduced by compression, thereby leading to more optimal model performance and (2) how noise might offer DP for communications, a concept crucial for ensuring data privacy in FL. Future studies could examine the relationship between the noise generated by lossy compression and its impact on DP guarantees. These areas all warrant subsequent exploration to increase the merits of including lossy compression in this technology.

IX. CONCLUSION

Our study demonstrates that EBLCs can mitigate communication overhead in FL client-server communications without compromising model accuracy. This is particularly significant in scenarios where bandwidth is constrained, as our algorithm achieves remarkable compression ratios and maintains inference accuracy within small relative error bounds. Looking ahead, the potential of FEDSZ extends beyond current applications. We envisage its integration into a broader array of FL frameworks and scenarios, potentially enhancing data privacy through DP techniques inspired by the noise characteristics inherent in lossy compression processes. Not only this, but FEDSZ as a last-step in the communication pipeline will work effectively with other existing compression techniques for FL such as gradient sparsification, pruning, and quantization. Moreover, the open-source availability of FEDSZ within the APPFL framework promises to allow for usage beyond the scope of this project.

ACKNOWLEDGMENT

This research was supported by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research (ASCR), under contract DE-AC02-06CH11357, and supported by the National Science Foundation (NSF) under Grant OAC-2003709, OAC-2104023, and OAC-2311875. We acknowledge the computing resources provided on Swing (operated by Laboratory Computing Resource Center at Argonne). During the time of this work GW was supported by a Churchill Scholarship.

REFERENCES

[1] J. Wen, Z. Zhang, Y. Lan, Z. Cui, J. Cai, and W. Zhang, "A survey on federated learning: challenges and applications," *International Journal of Machine Learning and Cybernetics*, vol. 14, no. 2, pp. 513–535, 2023.

[2] C. Yang, M. Xu, Q. Wang, Z. Chen, K. Huang, Y. Ma, K. Bian, G. Huang, Y. Liu, X. Jin, and X. Liu, "Flash: Heterogeneity-aware federated learning at scale," *IEEE Transactions on Mobile Computing*, pp. 1–18, 2022.

[3] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečnỳ, S. Mazzocchi, B. McMahan, *et al.*, "Towards federated learning at scale: System design," *Proceedings of machine learning and systems*, vol. 1, pp. 374–388, 2019.

[4] E. Gabrielli, G. Pica, and G. Tolomei, "A survey on decentralized federated learning," *arXiv preprint arXiv:2308.04604*, 2023.

[5] A. Biswas and H.-C. Wang, "Autonomous vehicles enabled by the integration of iot, edge intelligence, 5g, and blockchain," *Sensors*, vol. 23, no. 4, p. 1963, 2023.

[6] J. Ding, E. Tramel, A. K. Sahu, S. Wu, S. Avestimehr, and T. Zhang, "Federated learning challenges and opportunities: An outlook," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8752–8756, IEEE, 2022.

[7] X. Liang, K. Zhao, S. Di, S. Li, R. Underwood, A. M. Gok, J. Tian, J. Deng, J. C. Calhoun, D. Tao, Z. Chen, and F. Cappello, "Sz3: A modular framework for composing prediction-based error-bounded lossy compressors," *IEEE Transactions on Big Data*, vol. 9, no. 2, pp. 485–498, 2023.

[8] K. Zhao, S. Di, M. Dmitriev, T.-L. D. Tonellot, Z. Chen, and F. Cappello, "Optimizing error-bounded lossy compression for scientific data by dynamic spline interpolation," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pp. 1643–1654, 2021.

[9] P. Lindstrom, "Fixed-rate compressed floating-point arrays," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 2674–2683, 2014.

[10] M. Ryu, Y. Kim, K. Kim, and R. K. Madduri, "Appfl: Open-source software framework for privacy-preserving federated learning," in *2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, (Los Alamitos, CA, USA), pp. 1074–1083, IEEE Computer Society, jun 2022.

[11] X. Liang, S. Di, D. Tao, S. Li, S. Li, H. Guo, Z. Chen, and F. Cappello, "Error-controlled lossy compression optimized for high compression ratios of scientific datasets," in *2018 IEEE International Conference on Big Data (Big Data)*, pp. 438–447, 2018.

[12] X. Yu, S. Di, K. Zhao, J. Tian, D. Tao, X. Liang, and F. Cappello, "Ultrafast error-bounded lossy compression for scientific datasets," in *Proceedings of the 31st International Symposium on High-Performance Parallel and Distributed Computing*, HPDC '22, (New York, NY, USA), p. 159–171, Association for Computing Machinery, 2022.

[13] X. Liang, S. Di, D. Tao, S. Li, B. Nicolae, Z. Chen, and F. Cappello, "Improving performance of data dumping with lossy compression for scientific simulation," in *2019 IEEE International Conference on Cluster Computing (CLUSTER)*, pp. 1–11, 2019.

[14] Y. Liu, S. Di, K. Chard, I. Foster, and F. Cappello, "Optimizing scientific data transfer on globus with error-bounded lossy compression," in *43rd IEEE International Conference on Distributed Computing Systems (IEEE ICDCS2023)*, 2023.

[15] T. Serra, A. Kumar, and S. Ramalingam, "Lossless compression of deep neural networks," 2020.

Authorized licensed use limited to: CLEMSON UNIVERSITY. Downloaded on September 19,2024 at 02:45:08 UTC from IEEE Xplore. Restrictions apply.

[16] S. Jin, S. Di, X. Liang, J. Tian, D. Tao, and F. Cappello, "Deepsz: A novel framework to compress deep neural networks by using error-bounded lossy compression," in *Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing*, HPDC '19, (New York, NY, USA), p. 159–170, Association for Computing Machinery, 2019.

[17] Z. Hu, X. Zou, W. Xia, S. Jin, D. Tao, Y. Liu, W. Zhang, and Z. Zhang, "Delta-dnn: Efficiently compressing deep neural networks via exploiting floats similarity," in *Proceedings of the 49th International Conference on Parallel Processing*, ICPP '20, (New York, NY, USA), Association for Computing Machinery, 2020.

[18] Y. Choi, M. El-Khamy, and J. Lee, "Universal deep neural network compression," *IEEE Journal of Selected Topics in Signal Processing*, vol. 14, no. 4, pp. 715–726, 2020.

[19] N. Ström, "Scalable distributed dnn training using commodity gpu cloud computing," 2015.

[20] N. Dryden, T. Moon, S. A. Jacobs, and B. Van Essen, "Communication quantization for data-parallel training of deep neural networks," in *2016 2nd Workshop on Machine Learning in HPC Environments (MLHPC)*, pp. 1–8, IEEE, 2016.

[21] C.-Y. Chen, J. Choi, D. Brand, A. Agrawal, W. Zhang, and K. Gopalakrishnan, "Adacomp: Adaptive residual gradient compression for data-parallel distributed training," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, 2018.

[22] A. F. Aji and K. Heafield, "Sparse communication for distributed gradient descent," *arXiv preprint arXiv:1704.05021*, 2017.

[23] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," *arXiv preprint arXiv:1712.01887*, 2017.

[24] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Sparse binary compression: Towards distributed deep learning with minimal communication," in *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, 2019.

[25] C. Renggli, S. Ashkboos, M. Aghagolzadeh, D. Alistarh, and T. Hoefler, "Sparcml: High-performance sparse communication for machine learning," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–15, 2019.

[26] F. Seide, H. Fu, J. Droppo, G. Li, and D. Yu, "1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns," in *Fifteenth annual conference of the international speech communication association*, 2014.

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

[28] W. Wen, C. Xu, F. Yan, C. Wu, Y. Wang, Y. Chen, and H. Li, "Terngrad: Ternary gradients to reduce communication in distributed deep learning," *Advances in neural information processing systems*, vol. 30, 2017.

[29] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "Qsgd: Communication-efficient sgd via gradient quantization and encoding," *Advances in neural information processing systems*, vol. 30, 2017.

[30] C. H. Martin and M. W. Mahoney, "Implicit self-regularization in deep neural networks: Evidence from random matrix theory and implications for learning," *The Journal of Machine Learning Research*, vol. 22, no. 1, pp. 7479–7551, 2021.

[31] Y. Li, T. Ma, and H. Zhang, "Algorithmic regularization in over-parameterized matrix sensing and neural networks with quadratic activations," in *Conference On Learning Theory*, pp. 2–47, PMLR, 2018.

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

[33] M. Yu, Z. Lin, K. Narra, S. Li, Y. Li, N. S. Kim, A. Schwing, M. Annavaram, and S. Avestimehr, "Gradiveq: Vector quantization for bandwidth-efficient gradient aggregation in distributed cnn training," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[34] H. Wang, S. Sievert, S. Liu, Z. Charles, D. Papailiopoulos, and S. Wright, "Atomo: Communication-efficient learning via atomic sparsification," *Advances in neural information processing systems*, vol. 31, 2018.

[35] J. P. Mellado, S. Sarkar, and Y. Zhou, " Large-eddy simulation of Rayleigh-Taylor turbulence with compressible miscible fluids," *Physics of Fluids*, vol. 17, p. 076101, 07 2005.

[36] K. Zhao, S. Di, X. Lian, S. Li, D. Tao, J. Bessac, Z. Chen, and F. Cappello, "Sdrbench: Scientific data reduction benchmark for lossy compressors," in *2020 IEEE International Conference on Big Data (Big Data)*, pp. 2716–2724, 2020.

[37] J. Liu, S. Di, S. Jin, K. Zhao, X. Liang, Z. Chen, and F. Cappello, "Srn-sz: Deep leaning-based scientific error-bounded lossy compression with super-resolution neural networks," 2023.

[38] K. C. Barr and K. Asanović, "Energy-aware lossless data compression," *ACM Trans. Comput. Syst.*, vol. 24, p. 250–291, aug 2006.

[39] S. Li, S. Di, X. Liang, Z. Chen, and F. Cappello, "Optimizing lossy compression with adjacent snapshots for n-body simulation data," in *2018 IEEE International Conference on Big Data (Big Data)*, pp. 428–437, 2018.

[40] D. Tao, S. Di, Z. Chen, and F. Cappello, "In-depth exploration of single-snapshot lossy compression techniques for n-body simulations," in *2017 IEEE International Conference on Big Data (Big Data)*, (Los Alamitos, CA, USA), pp. 486–493, IEEE Computer Society, dec 2017.

[41] K. Zhao, S. Di, D. Perez, X. Liang, Z. Chen, and F. Cappello, "Mdz: An efficient error-bounded lossy compressor for molecular dynamics," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pp. 27–40, 2022.

[42] Blosc Development Team, "A fast, compressed and persistent data store library," 2009-2023. https://blosc.org.

[43] Gailly, Jean-loup and Adler, Mark, "A Massively Spiffy Yet Delicately Unobtrusive Compression Library," 1995-2023. https://zlib.net/.

[44] Meta, "Zstandard." https://pypi.org/project/zstd/, 2023.

[45] Gailly, Jean-loup and Adler, Mark, "Gnu gzip," 1992-2023. https://www.gnu.org/software/gzip/.

[46] The Tukaani Project, "Xz utils," 2009-2023. https://tukaani.org/xz/.

[47] L. Dalcin and Y.-L. L. Fang, "mpi4py: Status update after 12 years of development," *Computing in Science & Engineering*, vol. 23, no. 4, pp. 47–54, 2021.

[48] T.-H. Hoang, J. Fuhrman, R. Madduri, M. Li, P. Chaturvedi, Z. Li, K. Kim, M. Ryu, R. Chard, E. Huerta, *et al.*, "Enabling end-to-end secure federated learning in biomedical research on heterogeneous computing environments with appflx," *arXiv preprint arXiv:2312.08701*, 2023.

[49] S. Bose and K. Kim, "Federated short-term load forecasting with personalization layers for heterogeneous clients," *arXiv preprint arXiv:2309.13194*, 2023.

[50] S. Bose, Y. Zhang, and K. Kim, "Privacy-preserving load forecasting via personalized model obfuscation," *arXiv preprint arXiv:2312.00036*, 2023.

[51] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics* (A. Singh and J. Zhu, eds.), vol. 54 of *Proceedings of Machine Learning Research*, pp. 1273–1282, PMLR, 20–22 Apr 2017.

[52] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," 2019.

[53] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[54] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, (Red Hook, NY, USA), p. 1097–1105, Curran Associates Inc., 2012.

[55] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[56] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms," 2017.

[57] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories," in *2004 Conference on Computer Vision and Pattern Recognition Workshop*, pp. 178–178, 2004.

[58] R. Singh, J. Kovacs, and T. Kiss, "To offload or not? an analysis of big data offloading strategies from edge to cloud," in *2022 IEEE World AI IoT Congress (AIIoT)*, pp. 046–052, 2022.

[59] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3*, pp. 265–284, Springer, 2006.

[60] W.-N. Chen, D. Song, A. Ozgur, and P. Kairouz, "Privacy amplification via compression: Achieving the optimal privacy-accuracy-communication trade-off in distributed mean estimation," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

Authorized licensed use limited to: CLEMSON UNIVERSITY. Downloaded on September 19,2024 at 02:45:08 UTC from IEEE Xplore. Restrictions apply.