Table 5. Hyperparameters for ResNet-18/CIFAR-100 from (Sahu et al., 2021)

Parameter	Value
Number of workers	8
Optimizer	SGD with momentum
Global batch size	1024
Momentum	0.9
Post warmup LR	1.6
LR decay	/10 at epoch 150 and 250
LR warmup	Linear for 5 epochs, starting from 0.1
Epochs	300
Weight decay	10^{-4}

A BUCKET PRIORITIZATION

Considering the fact that communication buckets have different impacts on training performance, we modified the `L-GreCo` algorithm so that the last buckets in transmission order, corresponding to the earlier layers, were compressed more. This compensates for the compression error caused by picking lower compression parameters for the first buckets, i.e., the last layers. In practical terms, we have added linear priorities to the layers in Algorithm 1, multiplying the size of each layer by the index of the bucket the layer is communicated in. The profile of communicated elements per buckets is shown in the Figure 8. We observe the linear shift of higher compression ratios towards the last buckets. However, bucket prioritization performs worse than original `L-GreCo`. It means that the effect of the first big buckets transmission is higher than the effect of better compression of the last buckets.

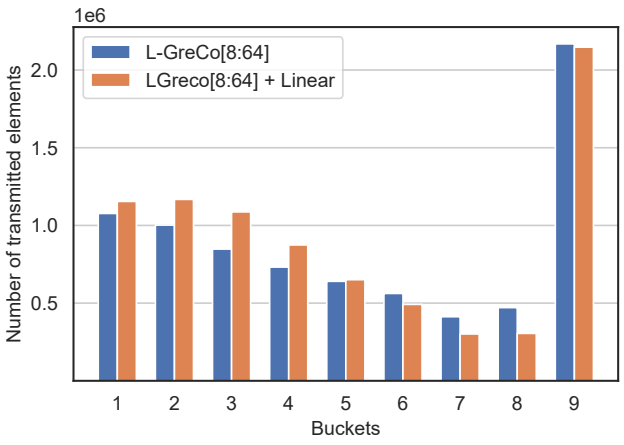


Figure 8. Communicated elements per bucket for L-GreCo and L-GreCo with linear bucket prioritizing. Transformer-XL with PowerSGD.

Table 6. Comparison of `L-GreCo` with other adaptive algorithms on ResNet-18/CIFAR-100.

Method	Parameters	Accuracy	Average Density (%)
Uniform	2%	71.8	2.00 (1×)
Uniform	0.1%	70.6	0.1 (20×)
Accordion	min = 0.1%, max = 2%	71.6	0.57 (3.5×)
Rethink-GS	$\lambda = 4.72 \times 10^{-3}$	71.4	0.35(5.7×)
L-GreCo	[0.1%, 10%]	71.7	0.30% (6.7×)

B LOW-RANK ERROR COMPUTATION

As discussed in Section 4, one of the main steps of our algorithm is to compute the error matrix for different possible compression parameters. Table 1 suggests that this is the most time-consuming part of our framework. Specifically for the PowerSGD, we need to compute low-rank errors for a wide range of ranks. There are two possible solutions to do so.

B.1 Singular Value Decomposition

The first way to compute errors is to use singular value decomposition and compute singular values for a particular layer, and calculate the approximation error for rank $r < min(m,n)$ by calculating $e_r = \sqrt{\sum_{i=r+1}^{min(m,n)} \sigma_i^2}$, which can be done efficiently for all ranks. Specifically, it is sufficient to compute squared singular values once, and then compute all the errors by a single matrix product. Thus, the bottleneck is computing singular values requiring $O(mn \cdot min(m,n))$ time and $O(n^2 + mn)$ space.

B.2 Power Iteration Steps

The second approach is to calculate the approximation error for each rank separately by doing a few power steps (without the communication parts); as Vogels et al. (2019) claims, this approach converges to the SVD-suggested matrix. On the practical side, we have observed that applying only 5 power steps is enough to have a small error relative to the optimal low-rank approximation suggested by SVD. This approach needs $O(mnr)$ time and $O((m+n) \cdot r)$ space for calculating rank r approximation error and therefore $O(mnr_{max}^2)$ to compute errors for all $r \in [r_{min}, r_{max}]$.

B.3 The Best of Both Worlds

Comparing computational complexity and memory requirements of two methods suggests it is better to use the power method when the rank range is small, e.g., ResNet50 on ImageNet or ResNet18 on Cifar100, and to use the SVD method when the rank range is large, e.g., TransformerXL and TransformerLM on WIKITEXT-103.

Table 7. Hyperparameters for ResNet-18/CIFAR-100

Parameter	Value
Number of workers	8
Optimizer	SGD with momentum
Global batch size	128
Momentum	0.9
Base LR	0.1
LR decay	/10 at epoch 150 and 250
Epochs	200
Weight decay	10^{-4}

C Combination of PowerSGD and L-GreCo

We note that in all of our wide-range experiments, the compression ratio when `L-GreCo` is applied to PowerSGD generally increases during the training (see Figure 9). This also aligns with the intuition behind the results of (Agarwal et al., 2021). This suggests that in this scenario, `L-GreCo` is able to increase the compression in the less crucial learning periods, e.g., last epochs.

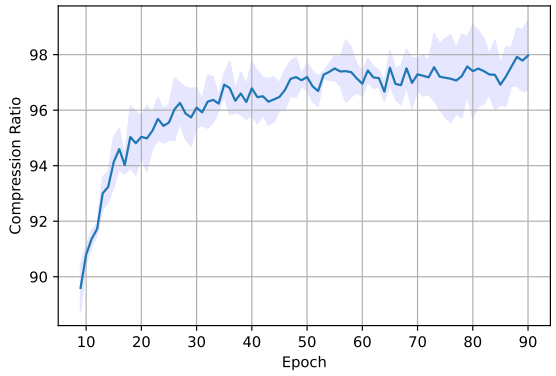


Figure 9. Compression ratio of the scheme suggested by `L-GreCo` during the training. ResNet50 with PowerSGD.

Table 8. Hyperparameters on ResNet-50/ImageNet

Parameter	Value
Number of workers	8
Optimizer	SGD with momentum
Global batch size	2048
Momentum	0.875
LR warmup	Linear for 8 epochs, starting from 0.256
LR schedule	cosine
LR decay	/10 at epoch 150 and 250
Epochs	90
Weight decay	1/32768
Label smoothing	0.1

Table 9. Hyperparameters on Transformer-XL/WikiText-103

Parameter	Value
Number of workers	8
Optimizer	LAMB
Global batch size	256
LR warmup	Linear for 1000 steps
LR schedule	cosine
Number of steps	40k
Weight decay	0.0

Table 10. Hyperparameters on Transformer-LM/WikiText-103

Parameter	Value
Number of workers	8
Optimizer	Adam
Adam betas	(0.9, 0.98)
Global batch size	2048
LR warmup	Linear for 4000 steps starting from 10^{-7}
LR schedule	inverse sqrt
Number of steps	50k
Weight decay	0.01

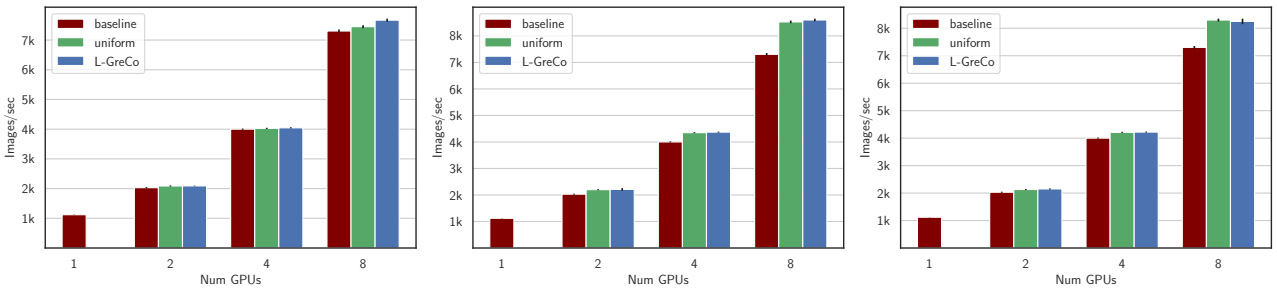
D Detailed Experimental Settings

For all the experiments, we used the standard hyperparameters, datasets, and data preprocessing. The detailed hyperparameters are shown in the tables 7, 8, 9, and 10.

For preprocessing the images of CIFAR-100 datasets, we follow the standard data augmentation and normalization routines. Random cropping and horizontal random flipping were applied for data augmentation. We also normalized each color with the following mean and standard deviation values for each channel: (0.4914, 0.4822, 0.4465) and (0.2023, 0.1994, 0.2010).

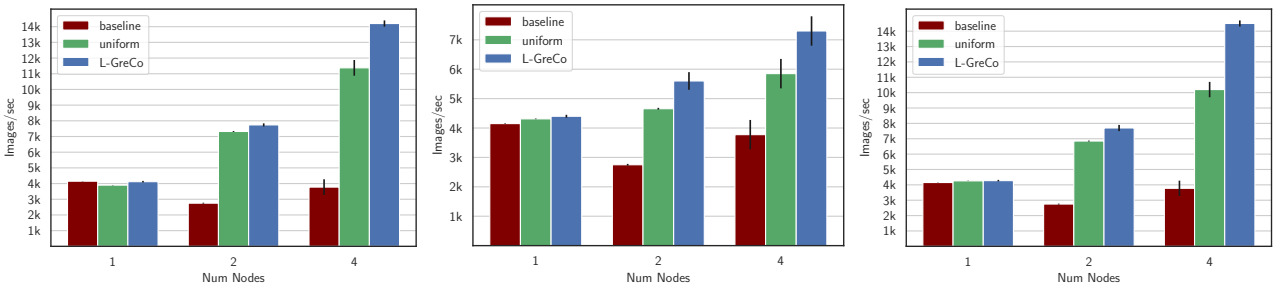
ResNet50 model uses the following data augmentation. We perform random resized crop to 224×224, scale from 8% to 100%, and do a random horizontal flip. Also, we do normalization with means (0.485, 0.456, 0.406) and standard deviations (0.229, 0.224, 0.225).

For wikitext-103 preprocessing, we used the standard preprocessing tools and tokenizers provided by Nvidia Examples (Nvidia, 2020) and FairSeq library (Ott et al., 2019).



(a) PowerSGD (b) QSGD (c) TopK

Figure 10. Throughput for ResNet50/ImageNet. Single node, RTX3090 GPUs.



(a) PowerSGD (b) QSGD (c) TopK

Figure 11. Throughput for ResNet50/ImageNet. Multi-node, each node has 4 RTX3090 GPUs.