Appendix of submission

Related Work

Federated/distributed learning compression approaches related to our work include Top-k sparsification-based approaches, Quantization-based approaches, and Hybrid approaches. Typical approaches of each are introduced in this section.

Top-k sparsification-based approaches: These approaches follow the idea that reducing the network overhead via uploading sparse local trained neural networks that only include k "important" parameters to the server. *Chen et al.* (Chen et al. 2018) proposed the Adaptive Residual Gradient Compression (AdaComp) scheme to automatically tune the compression ratio by dynamically deciding the maintained length of residual gradients. *Lin et al.* (Lin et al. 2018) uses momentum correction and local gradient clipping on the selected updates to ensure no loss of accuracy. Furthermore, DGC employs momentum factor masking and warmup training to mitigate the staleness problem caused by reduced communication. In this kind of approach, an common and important step is to accumulate those unupload update in local and add them to the next FL rounds, which have been proven to be essential to maintain the model performance during the training (Karimireddy et al. 2019; Alistarh et al. 2018; Stich, Cordonnier, and Jaggi 2018; Zheng, Huang, and Kwok 2019).

Quantization-based approaches: These approaches reduce communication overheads via encoding the uploaded parameters into fewer bits. Typically, *Seide et al.* (Seide et al. 2014) propose the 1-bit quantization technique. Each 32-bit floating-point parameter in a neural network is encoded into one bit to record its sign and then decoded with the sign and the mean value of the neural network's parameters. For keeping the accuracy and convergency of the resulting neural networks, the quantization error should be accumulated and added to the next round's encoding. *Wen et al.* (Wen et al. 2017) proposed a statistic-based stochastic algorithm, called TernGrad, to encode the transmitted floating-point gradients with two bits, while a theoretically-guaranteed convergence is proven. Alistarh *et al.* (Alistarh et al. 2017) extended the statistic-based quantization method by smoothly balancing the trade-off between communication bandwidth and convergency speed.

Hybrid approaches: *Strom et al.* (Strom 2015) combined the sparsification and 1-bit quantization techniques to compress the fully connected layers. Furthermore, they employed Golomb coding to "the index" part for further improving the compression ratio. Since sparsification and quantization are two orthometric techniques for compression, they are always applied together to achieve a higher compression ratio. *Dryden et al.* (Dryden et al. 2016) combined the sparsification and quantization but employed a percentage threshold in the sparsification and adopted the selected parameters' mean value as the quantized result. *Sattler et al.* (Sattler et al. 2020; Sattler et al. 2019) adopted sparsification and quantization techniques and introduced the Golomb encoding to reduce the cost of "the index" part, which provides additional $1.9\times$ compression in their work. *Tsuzuku et al.* (Tsuzuku, Imachi, and Akiba 2018) proposed a variance-based sparsification algorithm to compress the gradients in distributed deep learning. Additionally, 4-bit quantization and a Golomb encoding are further applied to improve the compression ratio.

Summarily, existing communication overheads reduction approaches in federated/distributed learning are mainly based on three orthogonal techniques: sparsification, quantization, and Golomb coding. Sparsification reduces the number of uploaded parameters; quantization further reduces the traffic overhead of those uploaded parameters; Golomb encoding minimizes the cost of "the index" part of those uploaded parameters (i.e., the selected parameters).

Comparisons among Different Parameter Selection Strategies

In the section 3 of main submission, we conduct the experiments about comparisons among different parameter selection strategies by training 4 datasets on GLnet*. In this section, additional experimental results on Vgg11* and Resnet18* are given.

Figure 1 shows that Kernel-based' selection performs an approximate FL model training results as 'Top-k', which are significantly better than the 'Rand-k' selection strategy. These results are consistent with the conclusion in Section 3 of the main submission.

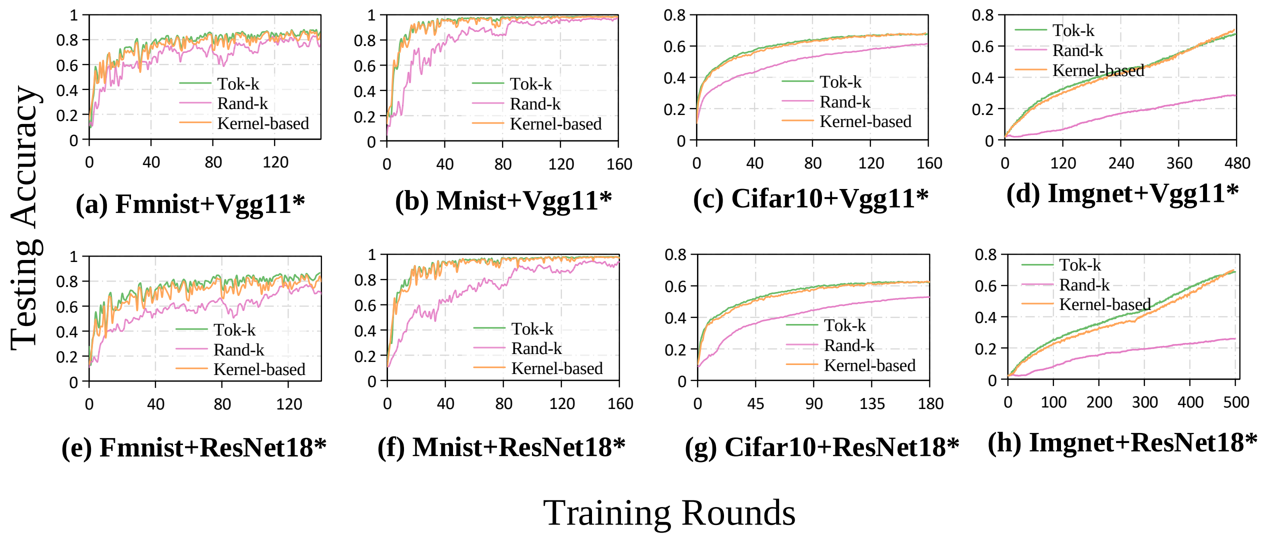


Figure 1: Comparisons of inference accuracy among three selecting strategies for FL compression

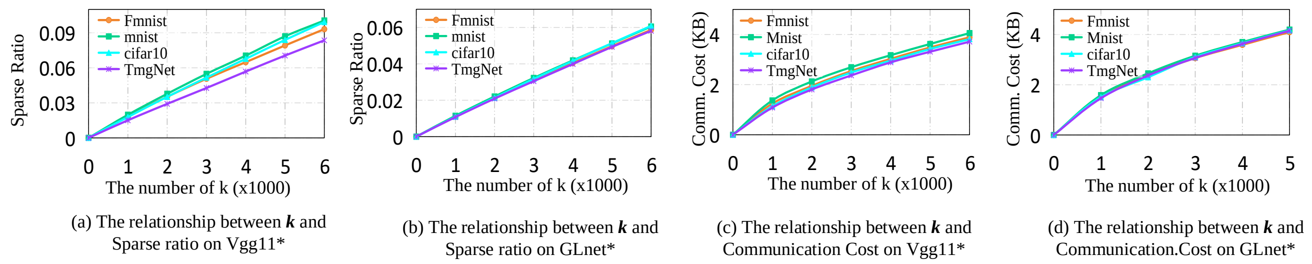


Figure 2: Comparisons of inference accuracy among threeselecting strategies for FL compression

Table 1: The specific model structure of Vgg11*, GLnet*, and Resnet18*.

CNNs	Vgg11*	GLnet	Resnet18
Layer 1	3×3 $Conv * 16$	7×7 $Conv * 24$	7×7 $Conv * 32$
Layer 2	Max pooling	max pooling	max pooling
Layer 3	3×3 $Conv * 32$	1×1 $Conv * 65$	Identity Block
Layer 4	Max pooling	3×3 $Conv * 128$	Identity Block
Layer 5	×3 $Conv * 64$	Max pooling	Conv Block
Layer 6	3×3 $Conv * 64$	Inception V1	Identity Block
Layer 7	Max pooling	Inception V1	Conv Block
Layer 8	3×3 $Conv * 128$	Max pooling	Identity Block
Layer 9	3×3 $Conv * 128$	Inception V1	Ave pooling
Layer 10	Max pooling	Ave pooling	FC layer
Layer 11	FC layer	FC layer	Output
Layer 12	Output	Output	

Supplementary of CNNs

the local Convolutional Neural Networks (CNNs) used for this work are structure and size simplified Vgg11 (Simonyan and Zisserman 2015), Resnet18 (He et al. 2016), and GoogleLeNet (Szegedy et al. 2015), which are named as Vgg11*, Resnet18*, GLnet*. The specific structures of Vgg11*, Resnet18*, and GLnet* are described in the Table 1. Specifically speaking, the simplifications are mainly three aspects. (1) Dropping several similar convolutional layers for a CNN. (2) Reducing the amount of convolution kernel for a convolutional layer. (3) Reducing the size of the fully connected layer. Therefore, the original characteristic of the CNN are not changed by above 3 operations.

The relationship of Hyper-parameter k, Sparse Ratio, and Communication Cost

In the section 5.1 of main submission, we conduct the experiments about the relationship among the hyper-parameter k, sparse ratio, and communication cost in SmartIdx by training 4 datasets on GLnet*. In this section, additional experimental results on Vgg11* and Resnet18* are given.

Figure 2 suggest that the both the sparse ratio ((a) and (b)) and communication cost ((c) and (d)) have almost achieved linear growth with the rising of hyper-parameter k. These results on different CNNs are similar to the results in section 5.1 of the main submission.

Table 2: Compression ratio with different sparse ratios of SBC, DGC, and SmartIdx on different CNNs and datasets.

CNNs	Methods	Fmnist			Mnist			Cifar10			TmgNet		
		SR	Acc	r	SR	Acc	r	SR	Acc	r	SR	Acc	r
VGG11*	SBC	16.2%	87.58%	165	14.9%	98.66%	186	18.6%	68.04%	178	17.9%	54.17%	499
	DGC	16.2%	88.14%	138	14.9%	98.77%	168	18.6%	68.05%	116	17.9%	64.28%	500
	SmartIdx	16.2%	88.01%	165	14.9%	98.70%	180	18.6%	68.24%	190	17.9%	70.60%	474
GLnet*	SBC*	15.9%	86.41%	165	16.9%	98.59%	190	28.0%	65.93%	189	17.6%	46.46%	499
	DGC*	15.9%	87.02%	142	16.9%	98.81%	176	28.0%	67.20%	145	17.6%	55.02%	451
	SmartIdx*	15.9%	87.02%	157	16.9%	98.81%	175	28.0%	67.21%	194	17.6%	55.21%	497
Resnet18*	SBC*	10.2%	86.13%	161	12.5%	98.22%	184	10.6%	63.01%	192	10.3%	47.50%	498
	DGC*	10.2%	86.13%	130	12.5%	98.22%	111	10.6%	63.12%	132	10.3%	60.97%	500
	SmartIdx*	10.2%	68.11%	162	12.5%	98.26%	173	10.6%	63.10%	186	10.3%	70.00%	496

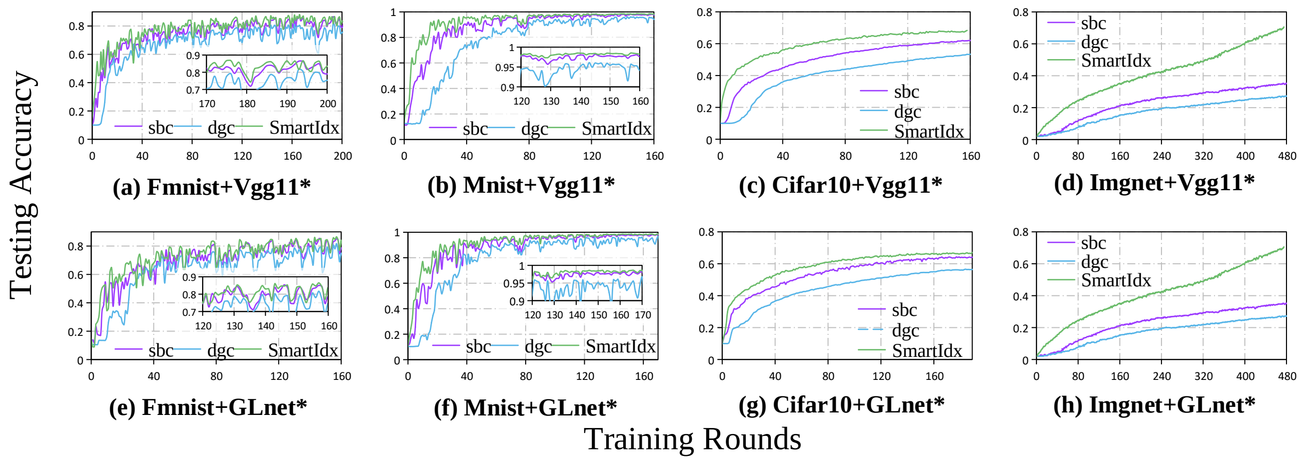


Figure 3: Training performance with the same communication costs of SBC, DGC, and SmartIdx on different datasets.

Details about Experiments with the Same Sparse Ratio

In the section 5.2 of main submission, we have compared the communication cost among SBC, DGC, and SmartIdx with the same sparse ratio. In this section, the specific sparse ratio, test accuracy, and training rounds related to the Table 2 of main submission are given.

Table 2 shows that in a few cases, SBC and DGC cannot reach the target accuracy within the required FL rounds, as suggested in the main submission. In other cases, the fastest convergence is always achieved by DGC as the momentum correction algorithm is employed in their approach, which efficiently improve model convergence. Note that the momentum correction algorithm is orthogonal technique to the compression framework of SmartIdx, thus it could be combined with SmartIdx for further optimizing the training performance.

Training performance With the Same Communication Cost

In the section 5.3 of main submission, we conduct the experiments about the Convergence Rounds with the Same Communication Cost by training 4 datasets on Resnet18*. In this section, additional experimental results on Vgg11* and GLnet* are given.

Figure 3 suggest that with the same communication cost per FL round, models trained by SmartIdx have higher test accuracy than SBC and DGC. This means that the superiority of SmartIdx exists in multiple configurations of CNNs and datasets. These performances are also similar to the results on section 5.3 of main submission.

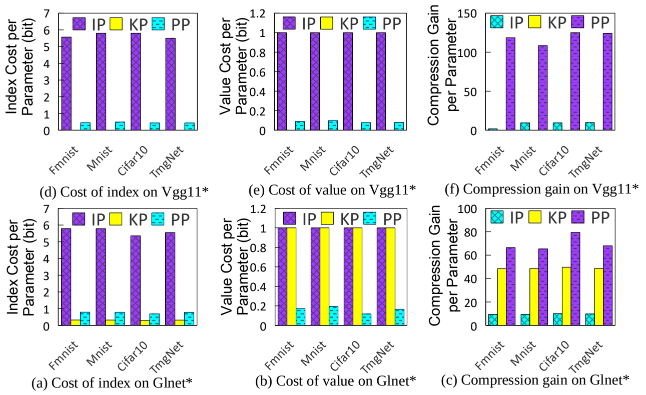
Comparison on Best Practices.

In the section 5.4 of main submission, we explore best practices among SBC, DGC, and SmartIdx by training Cifar10 and Tiny ImageNet on 3 different CNNs, respectively. In this section, additional experimental results of training Fmnist and Mnist on the same CNNs are given.

As can be seen the Table 3, the transmitting efficiency (σ) of our proposed SmartIdx are $2.51\times - 68.70\times$ than that on SBC and DGC in Fmnist and Mnist datasets, which means that most of the communication costs are consummed by value part but not index part. Meanwhile, SmartIdx achieves $2.75\times - 22.07\times$ higher compression ratio than the SBC and DGC in Fmnist and Mnist datasets. Note that compression ratio of DGC is the lowest due to there is no quantization

Table 3: Evaluations of compression ratio, transmitting efficiency, test accuracy and FL rounds on best practices.

Datasets	CNNs	R_b/Acc$_b$	SBC				DGC				SmartIdx			
			CR	σ	acc	r	CR	σ	acc	r	CR	σ	acc	r
Fmnist	VGG11*	(200/88%)	41.83	1.69	88.37%	162	16.49	0.17	88.19%	165	182.18	7.39	88.01%	162
	GLNet*	(200/87%)	20.14	1.51	88.06%	192	15.40	0.17	87.16%	162	126.86	4.53	87.30%	161
	ResNet18*	(200/86%)	59.29	1.83	86.00%	161	26.52	0.17	86.14%	162	412.81	10.55	86.08%	161
Mnist	VGG11*	(200/98.7%)	40.67	1.69	98.76%	165	14.84	0.17	98.70%	194	206.83	7.74	98.70%	180
	GLNet*	(200/98.8%)	39.65	1.84	98.79%	200	8.63	0.17	89.87%	184	109.17	4.62	98.81%	175
	ResNet18*	(200/98.2%)	108.98	1.36	98.20%	184	16.58	0.17	98.27%	175	365.87	11.45	98.26%	173



(d) Cost of index on Vgg11* (e) Cost of value on Vgg11* (f) Compression gain on Vgg11*

(a) Cost of index on Glnet* (b) Cost of value on Glnet* (c) Compression gain on Glnet*

Figure 4: Comparisons of index cost, value cost, and compression gain among different packages.

technique employed to the parameter values. Above superiority of best practices of SmartIdx are similar to the results on section 5.4 of main submission.

Details in each package

In the section 5.4 of main submission, we reveal the costs of index, the cost of value, and the Compression Gain in each package (i.e., *IP, KP, and PP*) by training 4 datasets on Resnet18*. In this section, the evaluation results on Vgg11* and GLnet* are given. Figure 4. (a) and (d) shows that the index costs consumed by *IP* are much larger than that on *KP* and *PP* since parameters in a convolution kernel share an index in *KP* and *PP*. Figure 4. (b) and (e) shows that the value costs consumed by *PP* are much lower than that on *IP* and *KP* since the convolution kernels belong to the same pattern share the same signs in the *PP* (Note that there is no parameter are selected into the kernel package in the Vgg11*, thus we denote it as zero). Figure 4. (c) and (f) shows that the the highest compression gain is achieved by *PP*, and the lowest CG is achieved by *IP*. Above evaluation results of each package are similar to the results on section 5.4 of main submission.

References

Alistarh, D.; Grubic, D.; Li, J.; and et al. 2017. QSGD: Communication-Efficient SGD via Gradient Quantization and Encoding. In *NeurIPS 2017, December 4-9, 2017, Long Beach, CA, USA*, 1709–1720.

Alistarh, D.; Hoefler, T.; Johansson, M.; and et al. 2018. The Convergence of Sparsified Gradient Methods. In *NeurIPS 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, 5977–5987.

Chen, C.; Choi, J.; Brand, D.; and et al. 2018. Ada-Comp : Adaptive Residual Gradient Compression for Data-Parallel Distributed Training. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, 2827–2835.

Dryden, N.; Moon, T.; Jacobs, S. A.; and Essen, B. V. 2016. Communication Quantization for Data-Parallel Training of Deep Neural Networks. In *2nd Workshop on Machine Learning in HPC Environments, MLHPC@SC, Salt Lake City, UT, USA, November 14, 2016*, 1–8.

He, K.; Zhang, X.; Ren, S.; and et al. 2016. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, 770–778. IEEE Computer Society.

Karimireddy, S. P.; Rebjock, Q.; Stich, S. U.; and et al. 2019. Error Feedback Fixes SignSGD and other Gradient Compression Schemes. In *ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, 3252–3261. PMLR.

Lin, Y.; Han, S.; Mao, H.; and et al. 2018. Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training. In *Proceedings of ICLR'18, Vancouver, Canada, April 2018*.

Sattler, F.; Wiedemann, S.; Müller, K.; and et al. 2019. Sparse Binary Compression: Towards Distributed Deep Learning with minimal Communication. In *Proceedings of IJCNN'19, Budapest, Hungary, July 2019*, 1–8.

Sattler, F.; Wiedemann, S.; Müller, K.; and et al. 2020. Robust and Communication-Efficient Federated Learning From Non-i.i.d. Data. *IEEE Transactions on Neural Networks and Learning Systems*, 31(9): 3400–3413.

Seide, F.; Fu, H.; Droppo, J.; and et al. 2014. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs. In *Proceedings of INTERSPEECH'15, Singapore, September 2014*.

Simonyan, K.; and Zisserman, A. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Stich, S. U.; Cordonnier, J.; and Jaggi, M. 2018. Sparsified SGD with Memory. In *NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, 4452–4463.

Strom, N. 2015. Scalable distributed DNN training using commodity GPU cloud computing. In *Proceedings of INTERSPEECH'15, Dresden, Germany, September, 2015*, 1488–1492.

Szegedy, C.; Liu, W.; Jia, Y.; and et al. 2015. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, 1–9.

Tsuzuku, Y.; Imachi, H.; and Akiba, T. 2018. Variance-based Gradient Compression for Efficient Distributed Deep Learning. In *Proceedings of ICLR'18, Vancouver, Canada, April 2018*.

Wen, W.; Xu, C.; Yan, F.; and et al. 2017. TernGrad: Ternary Gradients to Reduce Communication in Distributed Deep Learning. In *Proceedings of NIPS'17, Long Beach, CA, December 2017*, 1509–1519.

Zheng, S.; Huang, Z.; and Kwok, J. T. 2019. Communication-Efficient Distributed Blockwise Momentum SGD with Error-Feedback. In *NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, 11446–11456.