Term Paper - Communication Compression for Distributed Training

Naman Makkar
MEng Computer Science
Cornell Tech
nbm49@cornell.edu

Abstract

One of the greatest challenges faced in distributed training of deep neural networks is the communication bottleneck due to the frequent model updates transmitted across compute nodes. In order to alleviate these bottlenecks a variety of gradient compression techniques and algorithms have been utilised over the past few years which aim to minimise the decrease in accuracy caused by lossy compression while addressing the communication bottleneck. This term paper aims to provide a comprehensive survey of the gradient compression techniques that enhance the performance and efficiency of deep distributed training.

1 Introduction

Synchronous data parallel training is the most common and widely adopted approach for training modern over-parameterized neural networks like large language models and huge vision transformers. This approach requires combining per-node gradient updates at every iteration leading to an extremely high frequency of model updates transmitted across compute nodes. Communicating gradients accross compute nodes at a high frequency for a large parameter scale results in sub-optimal scalability in distributed training. So far there are two solutions to this -

- Utilising huge batch sizes to reduce the frequency of gradient update communication across compute nodes per epoch of training. This approach is flawed due to the widely observed degradation in accuracy as a result of utilizing large batch sizes for training.

- The second solution is to utilise gradient compression techniques which aim to reduce the size of the gradient update using techniques like gradient sparsification, quantization and low-rank approximation.

This paper aims to provide a survey of the various gradient compression techniques utilised for the purpose of optimizing deep distributed training. We will first look at traditional lossy gradient compression techniques (sparsification, quantization) and discuss their drawbacks. We will then look at the **Power-SGD** [VKJ20] algorithm and how it addresses the lacunae of the traditional lossy compression techniques. The project will finally focus on techniques like **Accordion** [Aye+23] that aim to optimise the accuracy vs per-iteration speedup through adaptive gradient compression and adaptive batch sizes.

2 Traditional Lossy Gradient Compression Techniques -

2.1 Quantization -

Gradient quantization reduces the size of the gradient update to be communicated by quantizing it to a low bit precision. This technique can be used to compress the gradients by up to 32x.

2.2 QSGD - SGD with Gradient Quantization

In the paper [Ali+17] the researchers implement SGD with gradient quantization and conducted experiments with 2-bit, 4-bit and 8-bit quantizations and showed a massive reduction in the communication costs.

A theoretical analysis was also provided demonstrating that QSGD maintains the convergence properties of standard SGD, with only a modest increase in the number of iterations needed for convergence.

2.3 Quantization to 1 bit precision (Sign quantization) -

In the research paper [Sei+14] the researchers show that by utilising a 1-bit quantization (a 32x gradient compression) along with an error feedback mechanism to compensate for the loss of precision it is possible to train deep neural networks for speech recognition on a large vocabulary task.

The error feedback mechanism utilised along with an aggressive quantization helps in retaining the information lost due to quantization for the subsequent iterations. Error feedback mechanism has proven to be a useful tool for biased compressors and we will be exploring it in more detail when we analyse PowerSGD.

2.4 Sparsification -

One of the advantages of sparsification is that it can compress the gradient update communication to a much greater extent than gradient quantization since quantization can only reduce the size of the gradient update by 32x (1 bit in 32 bits). Sparsification can compress gradients upto 3 orders of magnitude greater than quantization [Shi+19].

[SCJ18] show that Top-K sparsification allows communication reduction by a factor of the dimension of the problem itself. The paper goes on to compare Top-K sparsification with QSGD and shows that Top-K sparsification allows for transmitting two orders of magnitude fewer bits than QSGD.

2.4.1 Top K Sparsification -

Top K Sparsification [Lin+20] technique reduces the size of the gradient update communicated across compute nodes in distributed training by pruning the gradient values smaller than a certain chosen threshold and preserving only the K largest gradients where K is a predetermined value, usually 1%. The threshold is chosen as the value of the Kth largest gradient and all gradients larger than it are preserved. The rationale behind this technique is that the largest gradients are the most important for model performance and the size of the gradient update can be reduced be pruning the gradients smaller than a selected threshold.

2.5 Drawbacks -

1. **Non Linearity** - Majority of the traditional lossy gradient compression methods like Top-K sparsification for are non-linear compressors and the gradients compressed by them cannot be added hierarchically hence preventing the current SGD algorithms from making use of all-reduce (which is utilised in most deep learning frameworks), but instead require a gather operation. This reduces the scalability of these approaches.
2. Even though sparsification techniques like Random-K sparsification can be implemented with All-Reduce, the overhead for random memory access during gradient compression and decompression is high which makes it slower than regular SGD.
3. As observed in [Shi+19], Random-K sparsification was unable to converge on the ImageNet dataset.

3 Power SGD -

The Power SGD algorithm is a linear, biased compressor which addresses the drawbacks of the traditional lossy gradient compression techniques like sparsification and quantization by carrying out a computationally efficient low-rank approximation of the gradients of a model and can be easily implemented with **all-reduce** making the approach scalable for application in distributed training.

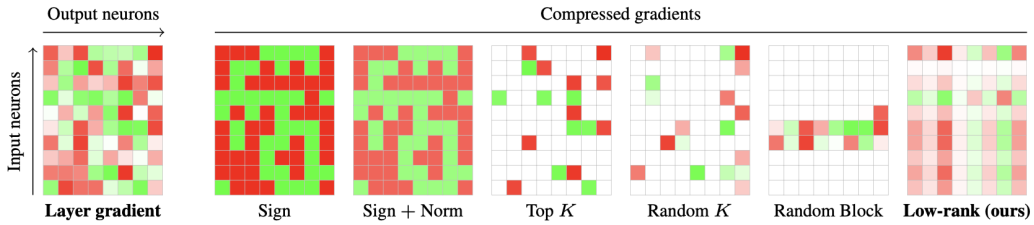

Figure 1: Figure taken from [VKJ20]. This figure visualizes the gradient of a layer on the left and its compressed representation as a matrix on the right using various gradient compression techniques such as sign quantization (1-bit quantization), top-K, random-K and random block sparsification and finally Power-SGD's low rank approximation. It can be observed by looking at the figure that the low-rank approximation calculated using Power-SGD is essentially carrying out a spectral regularization of the gradients of the layer [Gun+20].

3.1 The Low rank approximation algorithm

Algorithm 1 Rank-r POWERSGD compression

1: The update vector Δ_w is treated as a list of tensors corresponding to individual model parameters. Vector-shaped parameters (biases) are aggregated uncompressed. Other parameters are reshaped into matrices. The functions below operate on such matrices independently. For each matrix $M \in \mathbb{R}^{n\times m}$, a corresponding $Q \in \mathbb{R}^{m\times r}$ is initialized from an i.i.d. standard normal distribution.
2: **function** COMPRESS+AGGREGATE(update matrix $M \in \mathbb{R}^{n\times m}$, previous $Q \in \mathbb{R}^{m\times r}$)
3: $P \leftarrow MQ$
4: $P \leftarrow$ ALL REDUCE MEAN(P) ▷ Now, $P = \frac{1}{W}(M_1 + \ldots + M_W)Q$
5: $\hat{P} \leftarrow$ ORTHOGONALIZE(P) ▷ Orthonormal columns
6: $Q \leftarrow M^\top \hat{P}$
7: $Q \leftarrow$ ALL REDUCE MEAN(Q) ▷ Now, $Q = \frac{1}{W}(M_1 + \ldots + M_W)^\top \hat{P}$
8: **return** the compressed representation $(\hat{P}, Q)$.
9: **end function**
10: **function** DECOMPRESS($\hat{P} \in \mathbb{R}^{n\times r}$, $Q \in \mathbb{R}^{m\times r}$)
11: **return** $\hat{P}Q^\top$
12: **end function**

Figure 2: Figure taken from [VKJ20]. This gives an overview of the low rank approximation algorithm utilized in the Power SGD paper.

Low-rank approximation or rank-r approximation of the gradients of a model are calculated for each layer of the model independently. For each parameter gradient $M \in \mathbf{R^{nxm}}$ the rank-r approximation finds matrices $P \in \mathbf{R^{nxr}}$ and $Q \in \mathbf{R^{mxr}}$ where PQ^T is a rank-r approximation of the matrix M. This could also be calculated using SVD, however the low rank approximation algorithm utilised in Power-SGD is much more computationally efficient than SVD which has a computation complexity of $\mathbf{O(min(m,n)^2 * max(m,n))}$ which is not practical for over-parameterized model architectures.

Q is obtained from an i.i.d standard normal distribution, P is calculated using an all-reduce mean of the matrix product MQ over W workers. P is then orthogonalized using the Gram-Schmidt process which is the most expensive step of the rank-r compression. Q is then obtained using the orthogonalized matrix and the original gradient $M^T\hat{P}$. $\hat{P}$ and Q are obtained from the compression

step which can be decompressed by taking the matrix product $\hat{P}Q^T$ to get the rank-r approximation of M. The compression algorithm makes use of a 'warm-start' by reusing the approximation generated in the previous step.

3.2 Implementation with ALL-REDUCE

Algorithm 2 Distributed Error-feedback SGD with Momentum

hyperparameters: learning rate γ, momentum parameter λ
initialize model parameters $\mathbf{x} \in \mathbb{R}^d$, momentum $\mathbf{m} \leftarrow \mathbf{0} \in \mathbb{R}^d$, replicated across workers
at each worker $w = 1, \ldots, W$ **do**
  initialize memory $\mathbf{e}_w \leftarrow \mathbf{0} \in \mathbb{R}^d$
  for each iterate $t = 0, \ldots$ **do**
    Compute a stochastic gradient $\mathbf{g}_w \in \mathbb{R}^d$.
    $\Delta_w \leftarrow \mathbf{g}_w + \mathbf{e}_w$ ▷ Incorporate error-feedback into update
    $\mathcal{C}(\Delta_w) \leftarrow \text{COMPRESS}(\Delta_w)$
    $\mathbf{e}_w \leftarrow \Delta_w - \text{DECOMPRESS}(\mathcal{C}(\Delta_w))$ ▷ Memorize local errors
    $\mathcal{C}(\Delta) \leftarrow \text{AGGREGATE}(\mathcal{C}(\Delta_1), \ldots, \mathcal{C}(\Delta_W))$ ▷ Exchange gradients
    $\Delta' \leftarrow \text{DECOMPRESS}(\mathcal{C}(\Delta))$ ▷ Reconstruct an update $\in \mathbb{R}^d$
    $\mathbf{m} \leftarrow \lambda\mathbf{m} + \Delta'$
    $\mathbf{x} \leftarrow \mathbf{x} - \gamma\,(\Delta' + \mathbf{m})$
  end for
end at

Figure 3: Figure taken from [VKJ20]. This algorithm shows the implementation of Power-SGD with ALL-REDUCE in addition to an error feedback mechanism with the help of a post-compression momentum term $\mathbf{m}$ which ensures better convergence for biased compressors.

Since the Power-SGD is a linear compressor, it can be implemented with **ALL-REDUCE** without the use of a gather operation additionally since Power-SGD is a biased compressor it benefits from an error-feedback mechanism [Kar+19]. **Figure 3** shows the algorithm that implements Power-SGD with ALL-REDUCE and utilises a post-compression momentum term for implementing the Error-Feedback SGD. The Error feedback SGD first compresses the gradient update, after which the error feedback is calculated as the difference between the original gradient and the decompressed representation of the compressed gradient (i.e the rank-r approximation $\hat{P}Q^T$ of the original gradient). The compressed representations of the gradients (i.e $\hat{P}$ and Q) for all W workers are then aggregated using ALL-REDUCE mean.

3.3 Results and Observations

The Power-SGD algorithm was tested for image classification using a ResNet-18 on CIFAR-10 for rank 1 to rank 4 low rank approximation for gradient compression. For CIFAR-10 image classification, the Rank-1 approximation of PowerSGD achieved a reduction of 243x on the data communication per epoch while achieving a test accuracy of **93.6%** slightly lower than the regular SGD which achieved a test accuracy of **94.3%**. While the Rank-2 and Rank-4 approximations achieved a higher test accuracy than regular SGD while achieving a communication compression of **136x** and **72x** respectively.

Additionally, tests were carried out comparing the Power-SGD's biased compressor to an unbiased compressor (a compressor which after gradient compression and decompression yields the original gradient itself). In order to derive an unbiased compressor, a random matrix $U \in \mathbf{R^{mxr}}$ was sampled such that $\mathbf{E}[UU^T] = I_m$ and yielded low-rank approximation outputs (MU, U) such that after compression and decompression the original gradient could be reconstructed.

$$\mathbf{E}[(MU)U^T] = M\mathbf{E}[UU^T] = MI = M \tag{1}$$

It was observed that the Rank-1 Power-SGD and Rank-2 Power-SGD achieved accuracies of **93.6%** and **94.4%** on the CIFAR-10 test set while the unbiased Rank-1 and Rank-2 approximators only achieved accuracies of **71.2%** and **75.9%**. Both the biased and unbiased compressors achieved

similar amounts of data compression per epoch while the unbiased compressors got completely outperformed by the biased compressors, which is consistent with the findings of [Bez+22].

It was also observed that the best trade-off for model accuracy and per-iteration speed-up was provided as Rank-7 (for low-compression ratio and high performance) and Rank-2 (for a high-compression ratio and low performance).

3.4 Drawbacks -

1. Power-SGD does not address the trade-off between model accuracy and per-iteration speed-up. Due to its static compression ratio, training the model with a higher compression ratio (i.e a lower rank approximation) the model gives poor performance while for a lower compression ratio (i.e a higher-rank approximation) the algorithm doesn't provide a decently high per-iteration speedup for distributed training. Adaptive gradient compression techniques seek to address this issue.
2. The best rank or compression ratio either has to be derived experimentally or heuristically and could vary for each dataset.

4 Accordion Adaptive Gradient Compression -

Accordion is an adaptive gradient compression algorithm which adaptively implements gradient compression by recognising the critical regimes of model training. Making use of a low gradient compression rate for the critical training regimes and a high gradient compression rate for non-critical training regimes. This balances the trade-off between model accuracy and communication compression.

4.1 Identifying the critical regimes

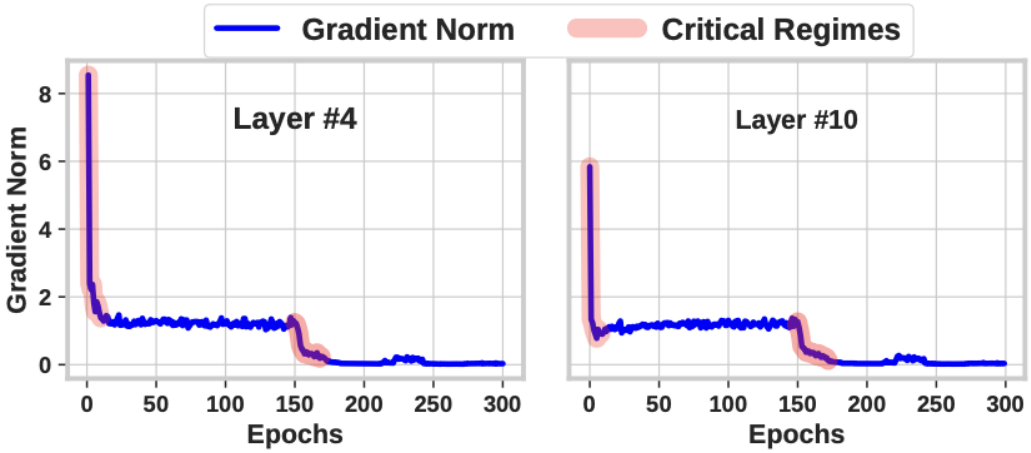

Figure 4: Figure taken from [Aye+23]. The critical regimes are identified by measuring the change in the gradient norm.

In order to tune the compression ratio, it is important to detect the critical regimes of training which is the most important epochs for the model training. Accordion detects the training regimes by detecting the rate of change of the gradient norms, avoiding the computationally expensive way of calculating the eigenvalues of the Hessian matrix to determine the critical training regimes of the model.

The following method was proposed for detecting the critical regimes of model training -

$$\frac{\|\Delta_{old}\| - \|\Delta_{curr}\|}{\|\Delta_{old}\|} \geq \eta \tag{2}$$

Where $\|\Delta_{curr}\|$ and $\|\Delta_{old}\|$ denote the accumulated gradient in the current and some previous epoch respectively, while η is the threshold used for declaring the critical regimes and is initialised to 0.5 for all the experiments conducted.

An interesting observation made by the researchers was that the critical regimes of model training occurred right at the beginning of the model training and after the learning rate decay has been carried out.

The Accordion algorithm switches between two compression ratios l_{low} and l_{high} based on whether the change in gradient norms according to equation 2 is higher or lower than the required threshold. In the context of PowerSGD l_{low} corresponds to a higher rank approximation and l_{high} corresponds to a lower rank approximation.

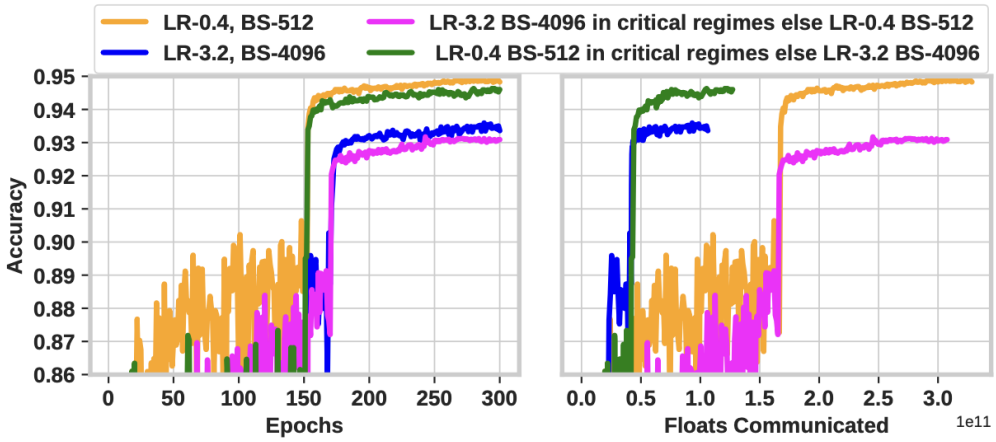

Figure 5: Figure taken from [Aye+23]. This figure explains the importance of utilising adaptive batch sizes. It can be observed that utilizing a lower batch size during the critical regimes and a higher batch size during the non-critical regimes achieves the best tradeoff between model accuracy and data communication compression. In the figure, the green plot utilises B_{low} for the critical regime of training while using B_{high} for the non-critical training regime, the yellow plot does not utilise adaptive batch sizes and communicates a significantly larger amount of data than the green plot while achieving roughly the same accuracy

Similarly, the Accordion algorithm also utilises adaptive batch sizes for model training. Utilising a higher batch size $B_{high} > B_{low}$ results in a reduction in communication by a factor of $\lfloor \frac{B_{high}}{B_{low}} \rfloor$

Therefore, in the critical regimes of training, Accordion utilises l_{low} gradient compression and B_{low} batch size and when in the non-critical training regime makes use of a larger batch size B_{high} and a high gradient compression ratio l_{high}

4.2 Accordion with PowerSGD -

Accordion applied with SGD using adaptive rank-r approximations on both CIFAR-10 and CIFAR-100 can be observed in **Figure 6**. It can be observed that adaptively switching between the higher and lower rank-r approximation allowed the adaptive gradient compression approach to achieve a data communication compression and runtim speedup which was much greater than the data communication compression achieved by the higher rank approximation while at the same time achieving a much better model performance than the lower rank approximation.

This tradeoff was one of the drawbacks of the vanilla PowerSGD algorithm and is very efficiently addressed by Accordion through adaptive gradient compression.

4.3 Accordion with TopK sparsification

Looking at **Figure 7** we can observe that utilising Accordion with adaptive TopK sparsification yields model performance comparable to the model with low sparsity while managing a data communication

Table 1. ACCORDION with POWERSGD on CIFAR-10

Network	Rank	Accuracy	Data Sent (Million Floats)		Time (Seconds)	
Resnet-18	Rank 2	**94.5%**	2418.4	(1×)	3509	(1×)
	Rank 1	94.1%	1350.4	(1.7×)	3386	(1.03×)
	ACCORDION	**94.5%**	1571.8	(**1.5**×)	3398	(**1.03**×)
VGG-19bn	Rank 4	**93.4%**	6752.0	(1×)	3613	(1×)
	Rank 1	68.6%	2074.9	(3.25×)	3158	(1.14×)
	ACCORDION	**92.9%**	2945.1	(**2.3**×)	3220	(**1.12**×)
Senet	Rank 4	**94.5%**	4361.3	(1×)	4689	(1×)
	Rank 1	94.2%	1392.6	(3.1×)	4134	(1.13×)
	ACCORDION	**94.5%**	2264.4	(**1.9**×)	4298	(**1.09**×)

Table 2. ACCORDION with POWERSGD on CIFAR-100

Network	Rank	Accuracy	Data Sent (Million Floats)		Time (Seconds)	
Resnet-18	Rank 2	**71.7%**	2426.3	(1×)	3521	(1×)
	Rank 1	70.0%	1355.7	(1.8×)	3388	(1.04×)
	ACCORDION	**71.8%**	1566.3	(**1.6**×)	3419	(**1.03**×)
DenseNet	Rank 2	**72.0%**	3387.4	(1×)	13613	(1×)
	Rank 1	71.6%	2155.6	(1.6×)	12977	(1.04×)
	ACCORDION	**72.5%**	2284.9	(**1.5**×)	13173	(**1.03**×)
Senet	Rank 2	**72.5%**	2878.1	(1×)	5217	(1×)
	Rank 1	71.5%	1683.1	(1.7×)	4994	(1.04×)
	ACCORDION	**72.4%**	2175.6	(**1.3**×)	5074	(**1.03**×)

Figure 6: Figure taken from [Aye+23]. This table shows a comparison of PowerSGD and Accordion algorithms on the CIFAR-10 and CIFAR-100 test set. It can be observed that Accordion with the help of adaptive gradient compression was able to achieve model performance equivalent to the higher rank approximation while achieving a much higher communication compression and training speedup.

Table 3. ACCORDION using TOPK on CIFAR-10

Network	K(%)	Accuracy	Data Sent (Billion Floats)		Time (Seconds)	
Resnet-18	99	**94.2%**	2626.1	(1×)	33672	(1×)
	10	93.1%	262.8	(9.9×)	7957	(4.2×)
	ACCORDION	**93.9%**	976.7	(**2.8**×)	9356	(**3.6**×)
GoogleNet	99	**94.6%**	1430.9	(1×)	28476	(1×)
	10	94.1%	145.2	(9.8×)	13111	(2.1×)
	ACCORDION	**94.7%**	383.8	(**3.7**×)	16022	(**1.7**×)
Senet	99	**94.6%**	2648.7	(1×)	29977	(1×)
	10	93.8%	267.9	(9.8×)	8055	(3.7×)
	ACCORDION	**94.5%**	869.8	(**3.0**×)	13071	(**2.29**×)

Table 4. ACCORDION using TOPK on CIFAR-100

Network	K(%)	Accuracy	Data Sent (Billion Floats)		Time (Seconds)	
Resnet-18	99	**72.4%**	2636.9	(1×)	53460	(1×)
	25	71.3%	659.4	(3.9×)	6176	(8.6×)
	ACCORDION	**72.3%**	923.6	(**2.8**×)	14223	(**3.8**×)
GoogleNet	99	**76.2%**	1452.4	(1×)	28579	(1×)
	25	75.3%	367.3	(3.9×)	12810	(2.23×)
	ACCORDION	**76.2%**	539.9	(**2.7**×)	15639	(**1.82**×)
Senet	99	**72.8%**	2659.5	(1×)	30312	(1×)
	25	71.9%	671.9	(3.9×)	7376	(4.1×)
	ACCORDION	**72.7%**	966.13	(**2.8**×)	10689	(**2.8**×)

Figure 7: Figure taken from [Aye+23]. This table shows a comparison of high and low TopK sparsification and Accordion algorithms on the CIFAR-10 and CIFAR-100 test set. It can be observed that Accordion with the help of adaptive gradient compression was able to achieve model performance equivalent to the training technique with higher gradient compression while achieving a much higher communication compression and training speedup.

compression comparable to the model with greater sparsity. This holds true for both the CIFAR-10 and CIFAR-100 datasets and all the models trained and tested on them.

4.4 Drawbacks of Accordion

1. The threshold for detecting critical regimes using the rate of change of the gradient norm was heuristically selected to be 0.5. We do not know if this threshold value is the optimal value or not and it could vary for various datasets.
2. The choice of the compression ratios l_{low} and l_{high} and batch sizes B_{low} and B_{high} have to be made heuristically as well and are left to the user to choose, requiring a lot of experiments to achieve the optimal combination for adaptive gradient compression.

References

[Sei+14] Frank Seide et al. "1-Bit Stochastic Gradient Descent and Application to Data-Parallel Distributed Training of Speech DNNs". In: *Interspeech 2014*. Sept. 2014. URL: https://www.microsoft.com/en-us/research/publication/1-bit-stochastic-gradient-descent-and-application-to-data-parallel-distributed-training-of-speech-dnns/.

[Ali+17] Dan Alistarh et al. "QSGD: Communication-Efficient SGD via Gradient Quantization and Encoding". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/6c340f25839e6acdc73414517203f5f0-Paper.pdf.

[SCJ18] Sebastian U Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. "Sparsified SGD with Memory". In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/b440509a0106086a67bc2ea9df0a1dab-Paper.pdf.

[Kar+19] Sai Praneeth Karimireddy et al. *Error Feedback Fixes SignSGD and other Gradient Compression Schemes*. 2019. arXiv: 1901.09847 [cs.LG].

[Shi+19] Shaohuai Shi et al. *Understanding Top-k Sparsification in Distributed Deep Learning*. 2019. arXiv: 1911.08772 [cs.LG].

[Gun+20] Suriya Gunasekar et al. *Characterizing Implicit Bias in Terms of Optimization Geometry*. 2020. arXiv: 1802.08246 [stat.ML].

[Lin+20] Yujun Lin et al. *Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training*. 2020. arXiv: 1712.01887 [cs.CV].

[VKJ20] Thijs Vogels, Sai Praneeth Karimireddy, and Martin Jaggi. *PowerSGD: Practical Low-Rank Gradient Compression for Distributed Optimization*. 2020. arXiv: 1905.13727 [cs.LG].

[Bez+22] Aleksandr Beznosikov et al. *On Biased Compression for Distributed Learning*. 2022. arXiv: 2002.12410 [cs.LG].

[Aye+23] Fadhel Ayed et al. *Accordion: A Communication-Aware Machine Learning Framework for Next Generation Networks*. 2023. arXiv: 2302.00623 [cs.NI].